INCREMENTAL SMOOTHING AND MAPPING

A Dissertation
Presented to
The Academic Faculty

by

Michael Kaess

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
December 2008

INCREMENTAL SMOOTHING AND MAPPING

Approved by:

Frank Dellaert, Advisor
College of Computing
Georgia Institute of Technology

Aaron F. Bobick
College of Computing
Georgia Institute of Technology

Henrik I. Christensen
College of Computing
Georgia Institute of Technology

John J. Leonard
Mechanical Eng. and CS & AI Lab
Massachusetts Institute of Technology

James M. Rehg
College of Computing
Georgia Institute of Technology

Date Approved: October 10, 2008

In memory of my mother.

ACKNOWLEDGEMENTS

I am deeply grateful to Frank Dellaert for being a patient advisor and for keeping me focused in my research. Without his ideas, his insistence on creating helpful visualizations, his feedback on academic writing and, not least his providing of funding, this work would simply not have been possible. I would also like to thank Ron Arkin for supporting my initial mapping work. Further, I would like to thank Henrik Christensen for providing very helpful feedback and job advise, as well as Aaron Bobick and James Rehg for valuable ideas on how to make this dissertation more general. Finally, I thank John Leonard for his support and for providing wonderful feedback.

I am also thankful to many researchers that I met during my graduate student life. In particular Ananth Ranganathan for many discussions that had significant influence on this work, Drew Steedly for his insights and for always looking out for me, and Udo Frese, Dirk Hähnel and Cyrill Stachniss for valuable discussions and for providing access to their software. I would also like to thank all my current and former lab mates for making work a pleasant experience, but especially Ananth, Christian, Gian Luca, Grant, Kai, Mingxuan, Richard, Sang Min, Zia, Alex, Endo, Eric and Lilia.

Many thanks to my friends that I always could count on in good and also bad times, especially to Herman, Ernest, Martin and Truc, as well as to Jens, Jörg, Martin and Timo for keeping in touch across the Atlantic.

Finally, I want to thank my father for his love and for supporting my educational endeavors. He taught me how to build, repair and, most importantly, just disassemble electrical and mechanical devices, which inspired in me the desire to understand how things work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

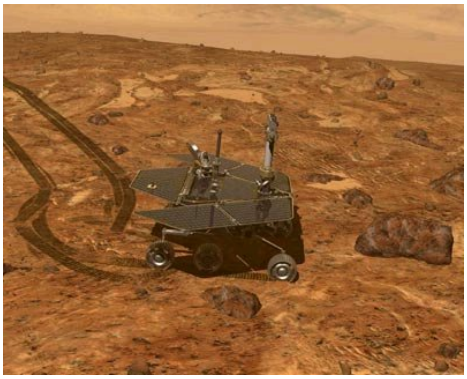
Incremental smoothing and mapping (iSAM) is presented, a novel approach to the simultaneous localization and mapping (SLAM) problem. SLAM is the problem of estimating an observer's position from local measurements only, while creating a consistent map of the environment. The problem is difficult because even very small errors in the local measurements accumulate over time and lead to large global errors. While SLAM is a key capability in mobile robotics today, the problem has a long history in land surveying. And the underlying estimation problem was already solved by Gauss in 1809 [56, 57] for computing the orbits of asteroids.

iSAM provides an exact and efficient solution to the SLAM estimation problem while also addressing data association. For the estimation problem, iSAM provides an exact solution by performing smoothing, which keeps all previous poses as part of the estimation problem, and therefore avoids linearization errors. iSAM uses methods from sparse linear algebra to provide an efficient incremental solution. In particular, iSAM deploys a direct equation solver based on QR matrix factorization of the naturally sparse smoothing information matrix. Instead of refactoring the matrix whenever new measurements arrive, only the entries of the factor matrix that actually change are calculated. iSAM is efficient even for robot trajectories with many loops as it performs periodic variable reordering to avoid unnecessary fill-in in the factor matrix. For the data association problem, I present state of the art data association techniques in the context of iSAM and present an efficient algorithm to obtain the necessary estimation uncertainties in real-time based on the factored information matrix. I systematically evaluate the components of iSAM as well as the overall algorithm using various simulated and real-world datasets.

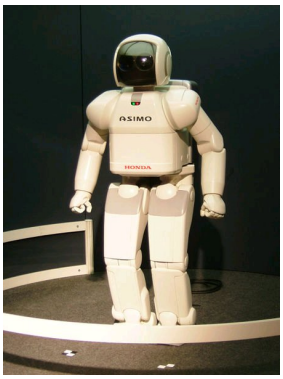
Chapter I

INTRODUCTION

(a) iRobot Roomba



(b) Mars Exploration Rover



(c) Honda Asimo

Figure 1: Examples of mobile robot applications that require or would benefit from real-time SLAM.

One of the main challenges for robots as they move out of factories is to find their way around less controlled environments to accomplish their assigned task. For many applications, such as service and entertainment robots (see Figure 1), it is infeasible to generate a model of the environment in advance. The key question then is how to integrate the robot's own local sensor data into a consistent "picture" of its environment. This is known as the problem of simultaneous localization and mapping (SLAM) [126, 91, 130]. A variety of SLAM algorithms have been presented, however, current solutions still have many shortcomings when it comes to practical application in large-scale environments. The three key issues are: 1) consistency, 2) computational complexity and 3) data association. Current algorithms typically only address one or two of these issues. In this work I present a novel approach to SLAM called incremental smoothing and mapping (iSAM) that addresses all of them at the same time.

1.1 Thesis

My thesis in this dissertation is the following:

> *Incremental smoothing and mapping (iSAM) provides a superior alternative to previous SLAM approaches that is exact, is efficient, supports data association by providing access to the exact marginal covariances, and is generally applicable to a wide range of SLAM problems.*

I split this thesis into four claims that correspond to the next four chapters of my dissertation. The goal of my research is to create a SLAM system called iSAM with the following properties:

1. *iSAM is exact:* No approximations are necessary and consistency is preserved even under the presence of nonlinear measurements.

2. *iSAM is efficient:* Incoming data are incorporated incrementally, so that the resulting algorithm is suitable for real-time application.

3. *iSAM supports data association:* Efficient access is provided to the exact marginal covariances that are needed for data association.

4. *iSAM is general:* iSAM works on simulated and real-world data ranging from laser-based landmark and pose-only applications to visual SLAM.

In the remainder of this chapter, I lay down the reasoning that leads to this thesis.

1.2 SAM: Smoothing and Mapping

Simultaneous localization and mapping (SLAM) is essential for mobile robots. Based on local sensor data only, a robot easily gets stuck in a cul-de-sac. In order to find an efficient way out of such a situation, the robot needs a map of its environment. However, in many settings it is impossible or simply too expensive to obtain such a map before deploying the robot. Therefore the robot itself needs to generate such a map from local sensor data. However, integrating noisy local sensor data into a consistent map is difficult because the

Figure 2: An example of 3D laser-based mapping [76] without SLAM. Note that the 3D model of the hallway curves to the left because of accumulated error from robot odometry.

accumulation of small local errors leads to larger global inconsistencies, see Figure 2 for an example. Creating a consistent map also requires localization in that map, and the combination of both is exactly what SLAM stands for.

Smoothing provides a superior solution to SLAM when compared to filtering methods. Nonlinear measurements are almost always encountered in mobile robot applications, typically through robot orientation or through bearing to landmarks. However, as filtering marginalizes out some variables, the linearization points of measurement equations connected to those variables become fixed and cannot be changed in the future. As a consequence linearization errors get introduced into the estimation process and eventually lead to catastrophic failure. Smoothing approaches in contrast can relinearize measurement equations at any time as they avoid marginalization and retain all variables. Therefore, smoothing provides qualitatively better solutions than filtering in the context of SLAM.

Smoothing is efficient as the underlying smoothing information matrix is sparse and remains sparse. The SLAM problem is naturally sparse, as it encodes the constraints between variables that are obtained by measurements. As a measurement typically connects only two variables, such as two robot poses or a robot pose and a landmark location, the resulting graphical structure of the problem is always sparse. Equivalently, the information matrix, which encodes this graph structure, is also sparse. This is in contrast to filtering, which yields constraints between all variables due to the marginalization process. Even though filtering only has to deal with a subset of the variables involved in the smoothing

process, for any non-trivial problem the number of entries in the dense filtering information matrix still grows much faster than for the corresponding sparse smoothing information matrix. Taking advantage of this sparse structure makes efficient solutions to smoothing possible.

1.3 iSAM: Incremental Smoothing and Mapping

In order to be useful for the operation of a mobile robot, SLAM needs to perform in real-time. Offline calculations are not an option for a mobile robot, as this implies stopping the robot until calculations finish. As the map obtained by a SLAM algorithm is directly used for robot navigation and planning, it is essential that an estimate based on all sensor data gathered until the current time is always available. Therefore, any SLAM algorithm has to perform in real-time in order to be useful for application on a mobile robot.

A real-time SLAM algorithm needs to incorporate new data incrementally, rather than operate in batch mode. New measurements for SLAM typically only affect a small part of the map. For example, receiving more data from inside an office of a large building generally does not have any influence on the mapping process for distant parts of the building. But a batch algorithm also recalculates the parts of the map that are unaffected. For real-time applications this is not an option, as it performs unnecessary computations that restrict the applicability of the algorithm to smaller environments. It is therefore essential that the SLAM algorithm incorporates new measurements incrementally, by building on previous calculations rather than starting from scratch after each step.

1.4 Data Association

Data association is an essential component of SLAM. When returning to a previously visited area of the environment it becomes necessary to identify this situation and correct any accumulated error in the robot's pose estimate. Such loop closing requires establishing correspondences between current and earlier observations, where earlier observations are summarized in the map. Without loop closing, errors will continue to accumulate until the map becomes practically useless. See Figure 3 for an example of a consistent map that required multiple loop closings. Data association is therefore an integral part of the SLAM

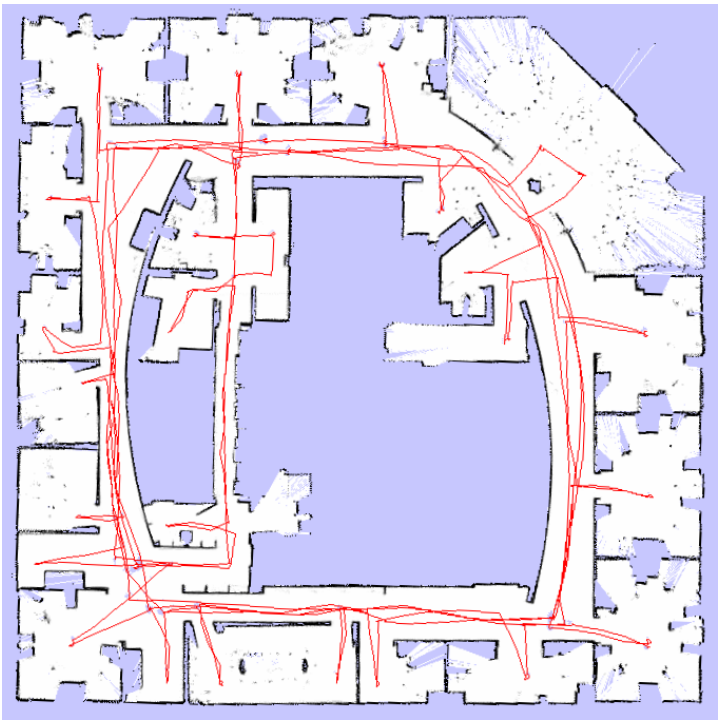


Figure 3: Generating a consistent map such as the one shown here requires a solution to the data association problem. For more details about this map see Section 3.5.2.

problem that cannot be ignored.

Access to the underlying uncertainties of the estimation process is needed for data association. Establishing correspondences requires deciding if a specific observation refers to a previously observed landmark or arises from a new landmark. In order to reduce both ambiguity and complexity, the search has to be restricted to a region around the estimated landmark location. However, the size and shape of this region depends on the estimation uncertainty underlying SLAM. Therefore, to perform an efficient search without discarding potential matches, knowledge of the exact uncertainties is essential for data association.

Identifying informative measurements is an important part of data association. Even if hundreds of measurements are available, using all of them will not necessarily improve the quality of the result. To the contrary, the large number of constraints might make it difficult to obtain a solution in real-time as they make the estimation problem more expensive to solve. Identifying which measurements significantly contribute to the accuracy of the result is therefore an important part of data association.

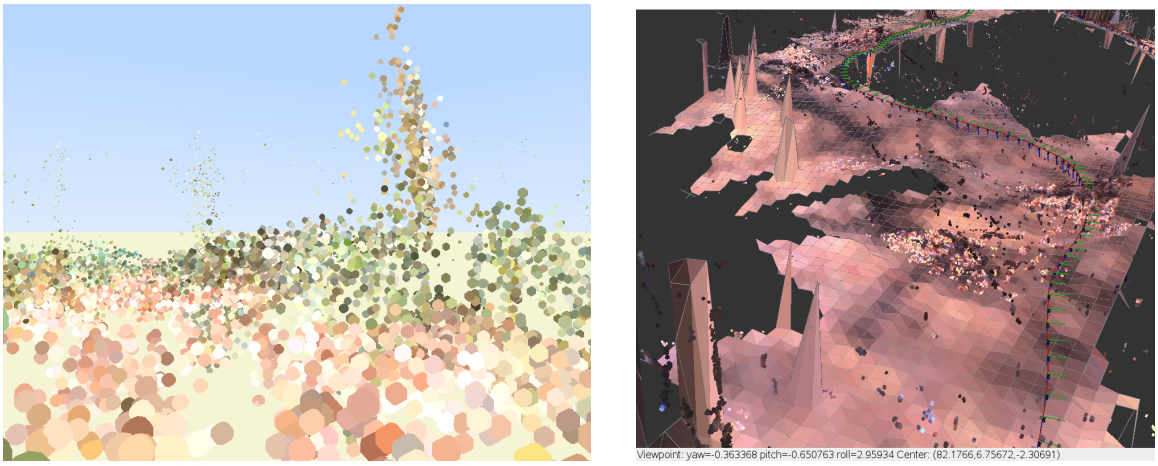


Figure 4: 3D maps of outdoor environments generated by visual SLAM; see Chapter 5 for more details.

1.5 General Applications including Visual SLAM

A SLAM algorithm needs to work for a wide range of SLAM settings. Different applications of SLAM include planar motion with three degrees of freedom as well as full spatial motion with six degrees of freedom. The application can require mapping 2D or 3D point landmarks, as well as more general objects such as lines or line segments. Furthermore, instead of landmarks, pose constraints might be used. And a variety of sensors with different noise characteristics can provide the data, from sonar over laser-based range sensors to cameras. As the underlying problems of estimation and data association are always the same, a successful solution to SLAM is also expected to work under any such setting.

Unstructured outdoor environments in combination with vision sensors provide a particularly challenging setup for evaluation of any SLAM system. Outdoor environments, such as the ones reconstructed in Figure 4, are challenging because they are unstructured and require modeling six degrees of freedom, which include multiple angular measurements that introduce significant nonlinearities. In combination with vision sensors the problem gets complicated further by the resulting nonlinear bearing measurements and the high depth uncertainty. Such a setting therefore provides a challenging test for any SLAM algorithm.

1.6 Organization

The remainder of this dissertation is organized as follows. Related work is discussed separately in each of the four main chapters. Chapter 2 presents our batch algorithm for smoothing and mapping (SAM) that provides an exact solution by avoiding any approximations as well as dealing correctly with nonlinear measurements. In Chapter 3 I present the incremental SAM (iSAM) algorithm that provides an efficient solution as required for real-time applications. Chapter 4 deals with the problem of data association and shows how iSAM provides the necessary information. In Chapter 5 I present an application of iSAM to the problem of visual SLAM. Chapter 6 concludes this dissertation and discusses future work.

Chapter II

SAM: SMOOTHING AND MAPPING

In this chapter I review the formulation of the SLAM problem in the context of smoothing, following Dellaert and Kaess [32], but using QR instead of Cholesky matrix factorization. Formulating SLAM in a smoothing context adds the complete trajectory into the estimation problem, thereby simplifying its solution. While this seems counter-intuitive at first, because more variables are added to the estimation problem, the simplification arises from the fact that the smoothing information matrix is naturally sparse. In contrast, in filtering approaches the information matrix becomes dense when marginalizing out robot poses. As a consequence of applying smoothing, we are able to provide an exact, yet efficient batch solution based on a sparse matrix factorization of the smoothing information matrix in combination with back-substitution. We call this matrix factor the *square root information matrix*, based on earlier work on square root information filtering (SRIF) and smoothing (SRIS), as recounted in [8, 96].

I start with the probabilistic model underlying the smoothing approach to SLAM. I then present an equivalent least-squares formulation of the smoothing problem. Linearization of the measurement functions leads to a standard linear least-squares problem. I continue with an efficient batch solution for the linear least-squares problem based on QR matrix factorization, and finally present results from applying SAM to a multi-camera dataset.

2.1 A Probabilistic Model for SLAM

I formulate the SLAM problem in terms of the *belief network* model shown in Figure 5. I denote the robot states by $X = \{\mathbf{x}_i\}$ with $i \in 0 \dots M$, the landmarks by $L = \{\mathbf{l}_j\}$ with $j \in 1 \dots N$, the control inputs by $U = \{\mathbf{u}_i\}$ for $i \in 1 \dots M$ and finally the landmark measurements by $Z = \{\mathbf{z}_k\}$ with $k \in 1 \dots K$. The joint probability of all variables and

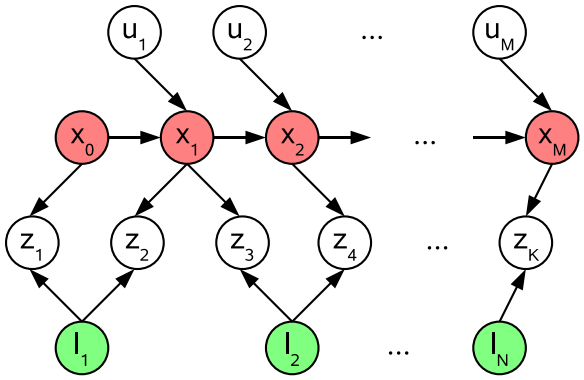


Figure 5: Bayesian belief network representation of the SLAM problem. Filled nodes represent variables: $\mathbf{x}_i$ is the state of the robot at time i, $\mathbf{l}_j$ the location of landmark j. The remaining nodes are measurements: $\mathbf{u}_i$ is the control input at time i and $\mathbf{z}_k$ the k^{th} landmark measurement.

measurements is given by

$$P(X, L, U, Z) \propto P(\mathbf{x}_0) \prod_{i=1}^{M} P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \prod_{k=1}^{K} P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \tag{1}$$

where $P(\mathbf{x}_0)$ is a prior on the initial state, $P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i)$ is the motion model, parametrized by the control input $\mathbf{u}_i$, and $P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k})$ is the landmark measurement model. Initially, I assume known correspondences (i_k, j_k) for each measurement $\mathbf{z}_k$. The problem of establishing correspondences, which is also called data association, is deferred to Chapter 4.

I assume Gaussian measurement models, as is standard in the SLAM literature [131]. The process model

$$\mathbf{x}_i = f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) + \mathbf{w}_i \tag{2}$$

describes the odometry sensor or scan-matching process, where $\mathbf{w}_i$ is normally distributed zero-mean process noise with covariance matrix Λ_i

$$P(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_i) \propto \exp -\frac{1}{2}||f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i||^2_{\Lambda_i} \tag{3}$$

where I use the notation $\|\mathbf{e}\|^2_\Sigma = \mathbf{e}^T\Sigma^{-1}\mathbf{e}$ for the squared Mahalanobis distance given a covariance matrix Σ. The Gaussian measurement equation

$$\mathbf{z}_k = h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) + \mathbf{v}_k \tag{4}$$

models the robot's landmark sensors, where $\mathbf{v}_k$ is normally distributed zero-mean measurement noise with covariance Γ_k

$$P(\mathbf{z}_k|\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \propto \exp -\frac{1}{2}||h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k||^2_{\Gamma_k}. \tag{5}$$

2.2 SLAM as a Least Squares Problem

To obtain an optimal estimate for the set of unknowns given all available measurements, I convert the SLAM problem into an equivalent least squares formulation based on a *maximum a posteriori* (MAP) estimate. As we perform smoothing rather than filtering, we are interested in the MAP estimate for the *entire trajectory* X and the map of landmarks L, given the control inputs U and the landmark measurements Z. The MAP estimate X^*, L^* for trajectory and map is obtained by minimizing the negative log of the joint probability from (1)

$$\begin{aligned} X^*, L^* &= \arg\max_{X,L} P(X, L, U, Z) \\ &= \arg\min_{X,L} -\log P(X, L, U, Z). \end{aligned} \tag{6}$$

Combined with the process and measurement models, this leads to the following nonlinear least squares problem

$$X^*, L^* = \arg\min_{X,L} \left\{ \sum_{i=1}^{M} \|f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i\|^2_{\Lambda_i} + \sum_{k=1}^{K} \|h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k\|^2_{\Gamma_k} \right\}. \tag{7}$$

Note that I have dropped the prior $P(x_0)$ on the first pose for simplicity.

If the process models f_i and measurement functions h_k are nonlinear and a good linearization point is not available, nonlinear optimization methods are used, such as Gauss-Newton or the Levenberg-Marquardt algorithm, which solve a succession of linear approximations to (7) to approach the minimum [33]. This is similar to the extended Kalman filter approach to SLAM as pioneered by [128], but allows for iterating multiple times to convergence, thus avoiding the problems arising from wrong linearization points.

2.3 Linearization

I review how to linearize the measurement functions and collect all components of the nonlinear least squares objective function (7) into one general least squares formulation.

We linearize the measurement functions in (7) by Taylor expansion, assuming that either a good linearization point is available or that we are working on one iteration of a nonlinear optimization method. In either case, the first-order linearization of the process term in (7) is given by

$$\begin{aligned} & f_i(\mathbf{x}_{i-1}, \mathbf{u}_i) - \mathbf{x}_i \\ \approx \; & \left\{ f_i(\mathbf{x}_{i-1}^0, \mathbf{u}_i) + F_i^{i-1} \delta \mathbf{x}_{i-1} \right\} - \left\{ \mathbf{x}_i^0 + \delta \mathbf{x}_i \right\} \\ = \; & \left\{ F_i^{i-1} \delta \mathbf{x}_{i-1} - \delta \mathbf{x}_i \right\} - \mathbf{a}_i \end{aligned} \tag{8}$$

where F_i^{i-1} is the Jacobian of the process model $f_i(.)$ at the linearization point $\mathbf{x}_{i-1}^0$, as defined by

$$F_i^{i-1} := \left. \frac{\partial f_i(\mathbf{x}_{i-1}, \mathbf{u}_i)}{\partial \mathbf{x}_{i-1}} \right|_{\mathbf{x}_{i-1}^0} \tag{9}$$

and $\mathbf{a}_i := \mathbf{x}_i^0 - f_i(\mathbf{x}_{i-1}^0, \mathbf{u}_i)$ is the odometry prediction error (note that $\mathbf{u}_i$ here is given and hence constant). The first-order linearizations of the measurement term in (7) are obtained similarly,

$$\begin{aligned} & h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k}) - \mathbf{z}_k \\ \approx \; & \left\{ h_k(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0) + H_k^{i_k} \delta \mathbf{x}_{i_k} + J_k^{j_k} \delta \mathbf{l}_{j_k} \right\} - \mathbf{z}_k \\ = \; & \left\{ H_k^{i_k} \delta \mathbf{x}_{i_k} + J_k^{j_k} \delta \mathbf{l}_{j_k} \right\} - \mathbf{c}_k \end{aligned} \tag{10}$$

where $H_k^{i_k}$ and $J_k^{j_k}$ are respectively the Jacobians of the measurement function $h_k(.)$ with respect to a change in $\mathbf{x}_{i_k}$ and $\mathbf{l}_{j_k}$ evaluated at the linearization point $(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)$

$$H_k^{i_k} := \left. \frac{\partial h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{x}_{i_k}} \right|_{(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)} \tag{11}$$

$$J_k^{j_k} := \left. \frac{\partial h_k(\mathbf{x}_{i_k}, \mathbf{l}_{j_k})}{\partial \mathbf{l}_{j_k}} \right|_{(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)} \tag{12}$$

and $\mathbf{c}_k := \mathbf{z}_k - h_k(\mathbf{x}_{i_k}^0, \mathbf{l}_{j_k}^0)$ is the measurement prediction error.

Using the linearized process and measurement models (8) and (10), respectively, (7) becomes

$$\begin{aligned} \delta \boldsymbol{\theta}^* = \arg\min_{\delta \boldsymbol{\theta}} \Bigg\{ & \sum_{i=1}^{M} \left\| F_i^{i-1} \delta \mathbf{x}_{i-1} + G_i^i \delta \mathbf{x}_i - \mathbf{a}_i \right\|_{\Lambda_i}^2 \\ & + \sum_{k=1}^{K} \left\| H_k^{i_k} \delta \mathbf{x}_{i_k} + J_k^{j_k} \delta \mathbf{l}_{j_k} - \mathbf{c}_k \right\|_{\Gamma_k}^2 \Bigg\} \end{aligned} \tag{13}$$

where we combined the pose and landmark variables into a single vector $\boldsymbol{\theta} \in \mathbb{R}^n$, where $n = Md_{\mathbf{x}} + Nd_{\mathbf{l}}$ with $d_{\mathbf{x}}$ and $d_{\mathbf{l}}$ the dimensions of the pose and landmark variables respectively. That is, we obtain a *linear* least squares problem in $\delta\boldsymbol{\theta}$ that needs to be solved efficiently. To avoid treating $\delta\mathbf{x}_i$ in a special way, we introduce the matrix $G_i^i = -I_{d_{\mathbf{x}} \times d_{\mathbf{x}}}$.

By a simple change of variables we can drop the covariance matrices Λ_i and Γ_k. With $\Sigma^{-1/2}$ the matrix square root of Σ, we rewrite the Mahalanobis norm as follows

$$\|\mathbf{e}\|_{\Sigma}^2 := \mathbf{e}^T\Sigma^{-1}\mathbf{e} = (\Sigma^{-T/2}\mathbf{e})^T(\Sigma^{-T/2}\mathbf{e}) = \left\|\Sigma^{-T/2}\mathbf{e}\right\|^2 \tag{14}$$

that is, we can always eliminate Λ_i from (13) by pre-multiplying F_i^{i-1}, G_i^i, and $\mathbf{a}_i$ in each term with $\Lambda_i^{-T/2}$, and similarly eliminate Γ_k from the measurement terms. For scalar measurements this simply means dividing each term by the measurement standard deviation. Next I assume that this has been done and drop the Mahalanobis notation.

Finally, after collecting the Jacobian matrices into one large but *sparse measurement Jacobian* $A \in \mathbb{R}^{m \times n}$ with m measurement rows, and the vectors $\mathbf{a}_i$ and $\mathbf{c}_k$ into one right-hand side vector $\mathbf{b} \in \mathbb{R}^m$, we obtain the following standard least squares problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|A\boldsymbol{\theta} - \mathbf{b}\|^2 \tag{15}$$

where I dropped the $\delta\cdot$ notation for simplicity.

Such sparse least squares systems are typically converted into an ordinary linear equation system by setting the derivative of $\|A\boldsymbol{\theta} - \mathbf{b}\|^2$ to 0, resulting in the so called normal equations $A^TA\boldsymbol{\theta} = A^Tb$. This equation system can be solved by Cholesky decomposition of A^TA. I use an alternative approach by applying QR factorization directly to the measurement Jacobian, which will be beneficial for incremental factorization in Chapter 3. Also, in contrast to Cholesky factorization this avoids having to calculate the information matrix A^TA with the associated squaring of the matrix condition number.

2.4 Solving by QR Matrix Factorization

I apply the standard QR matrix factorization to the measurement Jacobian A to solve the least squares problem (15). The QR factorization of the measurement Jacobian A is defined

as

$$A := Q \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{16}$$

where $R \in \mathbb{R}^{n \times n}$ is the upper triangular square root information matrix (note that the information matrix is given by $R^T R = A^T A$) and $Q \in \mathbb{R}^{m \times m}$ is an orthogonal matrix. I apply this factorization to the least squares problem (15):

$$\begin{aligned} \|A\boldsymbol{\theta} - \mathbf{b}\|^2 &= \left\| Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - \mathbf{b} \right\|^2 \\ &= \left\| Q^T Q \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - Q^T \mathbf{b} \right\|^2 \\ &= \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \boldsymbol{\theta} - \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} \right\|^2 \\ &= \|R\boldsymbol{\theta} - \mathbf{d}\|^2 + \|\mathbf{e}\|^2 \end{aligned} \tag{17}$$

where I define $[\mathbf{d}, \mathbf{e}]^T := Q^T \mathbf{b}$ with $\mathbf{d} \in \mathbb{R}^n$ and $\mathbf{e} \in \mathbb{R}^{m-n}$. Equation (17) becomes minimal if and only if $R\boldsymbol{\theta} = \mathbf{d}$, leaving the second term $\|\mathbf{e}\|^2$ as the residual of the least squares problem. Therefore, the QR factorization simplifies the least squares problem to a linear system with a single unique solution $\boldsymbol{\theta}^*$:

$$R\boldsymbol{\theta}^* = \mathbf{d}. \tag{18}$$

Most of the work for solving this equation system has already been done by the QR decomposition, because R is upper triangular, so simple back-substitution [60] can be used. The result is the least squares estimate $\boldsymbol{\theta}^*$ for the complete robot trajectory as well as the map, conditioned on all measurements.

2.5 Experiments and Results

In [32], in addition to simulation results, I have evaluated the nonlinear incremental version of SAM on a challenging, large-scale vision-based SLAM problem. The data was gathered by a mobile robot equipped with eight cameras traversing an indoor office environment,

see [78] for details. This is a challenging data-set, both because of the amount of data that needed to be dealt with, as well as the logistical and calibration issues that plague multi-camera rigs. In addition, dealing with visual features is complex and prone to failure.

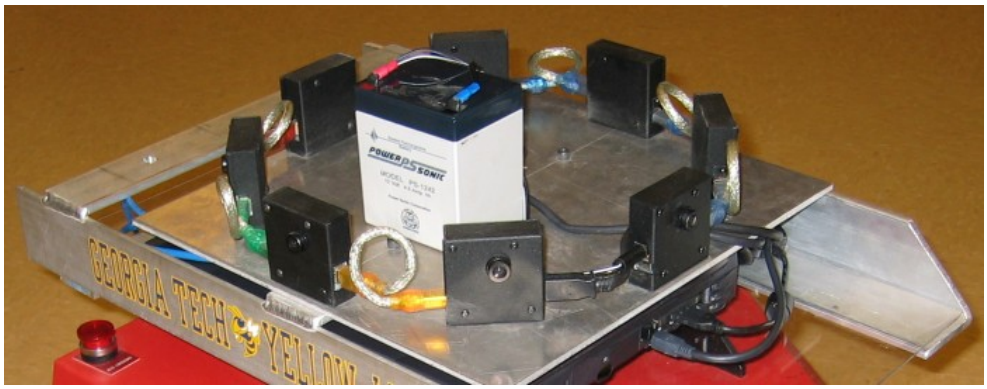


Figure 6: Custom made camera rig, mounted on top of an ATRV-Mini mobile robot. Eight Fire-Wire cameras are distributed equally along a circle and connected to an on-board laptop.

Figure 7: Combined snapshot from all eight cameras, providing a 360^o view. Note that the images do not fit perfectly as the cameras have different optical centers.

The measurements I use are features extracted from eight cameras mounted on top of an iRobot ATRV-Mini platform, as shown in Figure 6. They are matched between successive frames using RANSAC [45] based on a trifocal camera-arrangement. The data was taken in an office environment, with a bounding box of about $30m$ by $50m$ for the robot trajectory, and the landmarks sought are a set of unknown 3D points. The measurements consisted of the odometry provided by the robot, as well as 260 joint images, such as the one shown in Figure 7, taken with variable distances of up to $2m$ between successive views, and an overall trajectory length of about $190m$. The unknown poses were modeled as having 6 degrees of freedom (DOF), three translational and three rotational. Even though 3DOF seems sufficient for a planar indoor office environment, it turns out that 6DOF with a prior on pitch, roll and height is necessary, since any small bump in the floor has a clearly visible

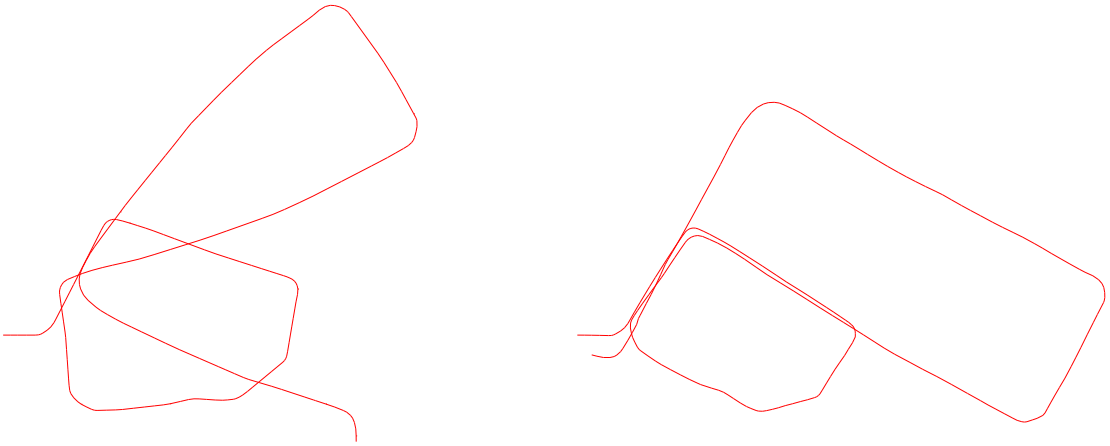


Figure 8: Left: Odometry as provided by the mobile robot platform. Right: Trajectory as reconstructed by our approach using visual input and odometry only. The trajectory has a length of about $190m$.

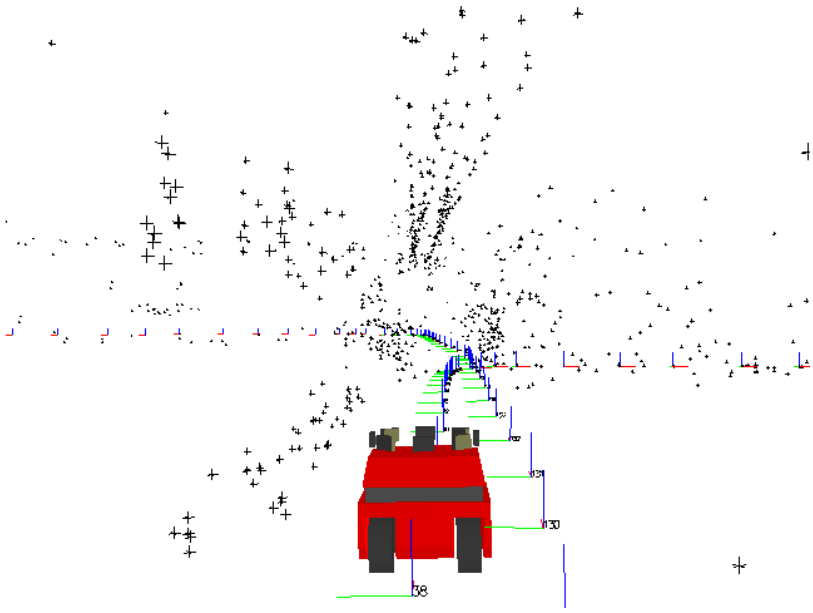
effect on the images. The standard deviations on x and y are $0.02m$ and on ϕ (yaw) $0.02rad$. The priors on z, θ (pitch) and ψ (roll) are all 0 with standard deviations $0.01m$ and $0.02rad$ respectively. The camera rig was calibrated in advance.

The SAM approach was able to deal with this large problem well within the real-time constraints of the application. My approach was to invoke the batch SAM algorithm after every three joint images taken by the robot. In each of these invocations, the problem was repeatedly re-linearized and factorized to yield an optimal update. I used sparse LDL [60] as the main factorization algorithm. More importantly, for ordering the variables I used COLAMD by Davis *et al.* [19] combined with the block-structured ordering heuristic.

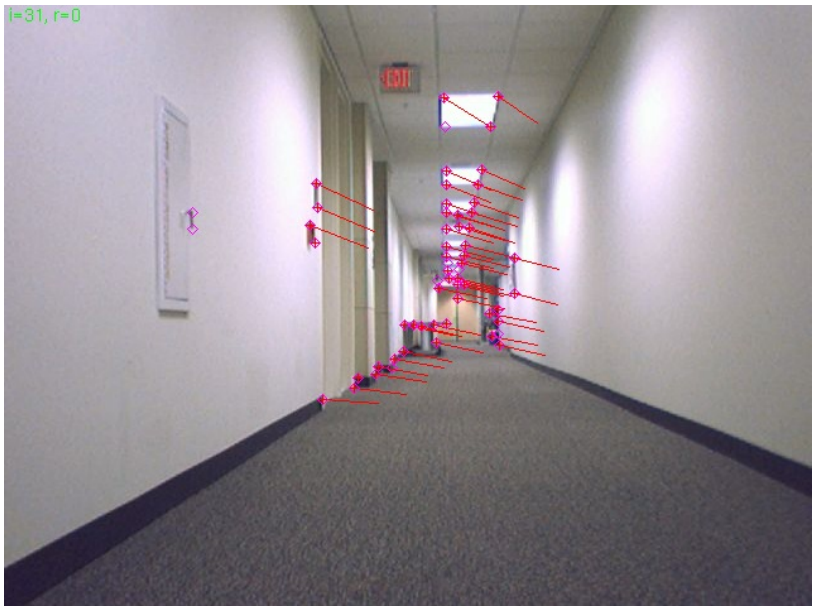
Processing the entire sequence took a total of 11 minutes and 10 seconds on a 2GHz Pentium-M based laptop, and the trajectory and map is shown in Figures 8 and 9. The correspondence matching yielded in 17 780 measurements on a total of 4383 unknown 3D points, taken from 260 different robot locations. In the final step, the measurement matrix A had 36 337 rows and 14 709 columns, with about $350K$ non-zero entries. Note that the map is three dimensional, which explains feature points in the middle of hallways, arising for example from overhead light fixtures as shown in Figure 10.

The relevant timing results are shown in Figure 11. The unexpected result is that *the factorization, because of the good variable ordering, is now but a minor cost in the whole*

Figure 9: Projected trajectory and map after applying batch SAM incrementally using visual input and odometry only. Each robot pose is shown as an outline of the ATRV-Mini platform. The recovered 3D structure is represented by green points. For comparison the manually aligned building map is shown in gray. The length of the trajectory is about $190m$. Given that no loop-closing was performed and considering the large scale of the environment and the incremental nature of the reconstruction method, this result is of very high quality. Note that features also occur along the ceiling, and that some features outside the building outline are caused by reflections.



(a) 3D model view.



(b) Camera image with measurements and reprojected structure.

Figure 10: Comparison of 3D model and actual image, for the data from Figure 9. (a) The 3D map as seen from inside a corridor showing the structure (black crosses), the robot coordinate systems (red,green,blue axes for x,y,z), and a 3D model of the robot. (b) A real image for comparison showing the features (magenta diamonds), image measurements (red crosses), and optical flow (red lines). In the 3D view, the lamps along the ceiling can be seen twice with a slight shift since the robot traversed the corridor twice and no loop closing is performed. Note that the error is small given the large scale of the environment.

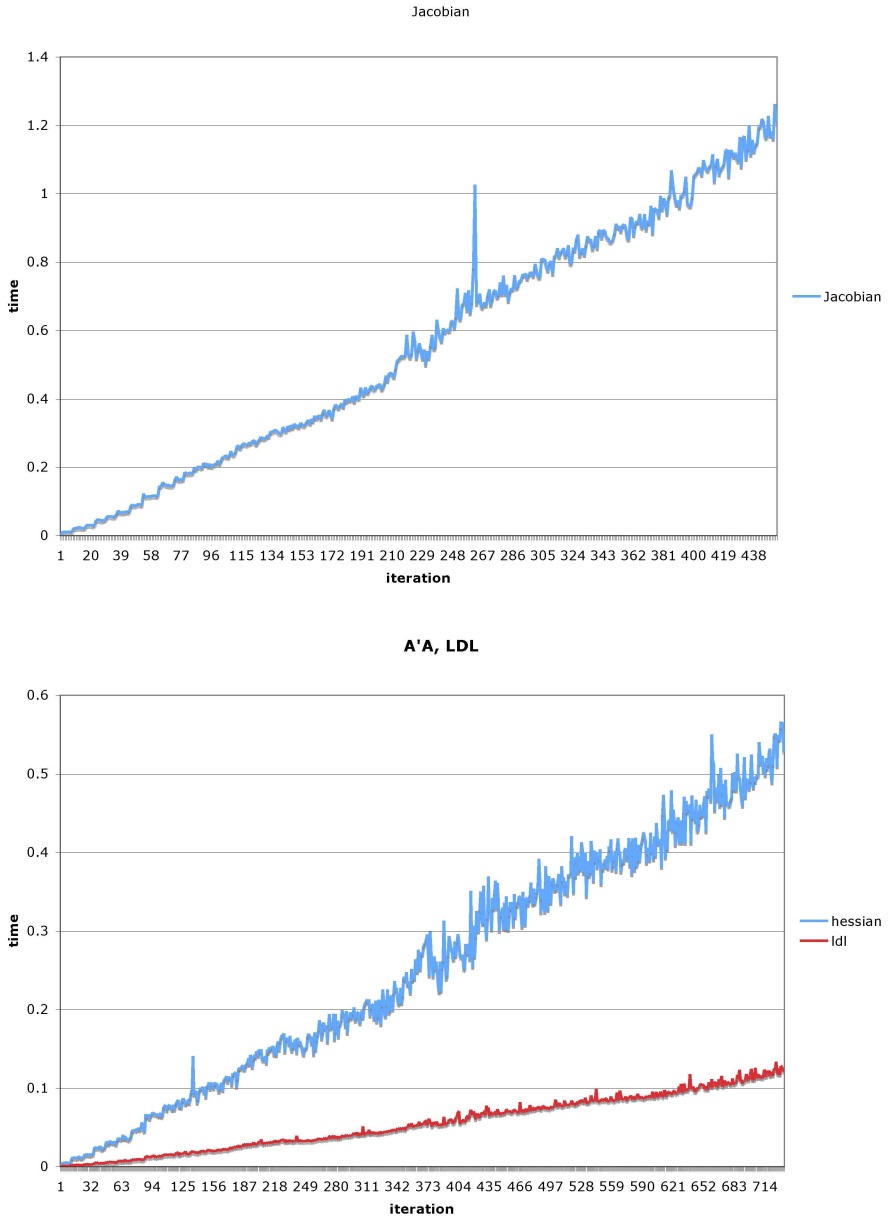


Figure 11: Timing results for batch SAM on the sequence from Figure 9. Top: The cost of forming the Jacobian A (linearization). Bottom: The cost for LDL factorization is less than for forming the Hessian or information matrix $\mathcal{I}$.

optimization. Instead, the largest cost is now evaluating the measurement Jacobian A (linearizing the measurement equations), which was done a total of 453 times. Its computational demands over time are shown in the panel at the top. Next in line is the computation of the information matrix $\mathcal{I} = A^T A$, shown above by "hessian" in the bottom panel. This is done exactly as many times as LDL itself, i.e., a total of 734 times. By the end of the sequence, this sparse multiplication (yielding - by then - a $15K \times 15K$ matrix) takes about 0.6 secs. In contrast, factorizing the resulting information matrix $\mathcal{I}$ takes just 0.1 seconds.

In the remaining part of this thesis, I completely avoid forming the information matrix by using a QR matrix factorization on the measurement Jacobian instead of the Cholesky factorization. This also allows for incremental factorizations that avoid the linear increase in computational complexity that is inherent to the batch process.

2.6 Related Work

There is a large body of literature on the field of robot localization and mapping, and I will only address closely related work as well as some of the most influential algorithms. A general overview of the area of SLAM can be found in [36, 2, 131, 130]. Initial work on probabilistic SLAM was based on the extended Kalman filter (EKF) and is due to Smith *et al.* [128], building on their earlier work [126, 127] as well as Durrant-Whyte's work [35]. Meanwhile it has been shown by Julier and Uhlmann [75] that filtering is inconsistent in nonlinear SLAM settings and much later work, such as [66, 85], focuses on reducing the effect of nonlinearities and providing more efficient, but typically approximate solutions to deal with larger environments.

Smoothing in the SLAM context avoids these problems by keeping the complete robot trajectory as part of the estimation problem. It is also called the *full* SLAM problem [131] and is closely related to bundle adjustment [134] in photogrammetry, and to structure from motion (SFM) [73] in computer vision. While those are typically solved by batch processing of all measurements, SLAM by nature is an incremental problem. Also, SLAM provides additional constraints in the form of odometry measurements and an ordered sequence of poses that are not present in general SFM problems. The first smoothing approach to the

SLAM problem by Lu and Milios [94] formulates the estimation problem as a network of constraints between robot poses. The first implementation by Gutmann and Nebel [68] was based on matrix inversion.

A number of improved and numerically more stable algorithms have since been developed, based on well-known iterative techniques such as relaxation [34, 11, 131], gradient descent [46, 47], conjugate gradient [86], and more recently multi-level relaxation [51, 53]. The latter is based on a general multi-grid approach that has proven very successful in other fields for solving systems of equations. While most of these approaches represent interesting solutions to the SLAM problem, they all have in common that it is very expensive to recover the uncertainties of the estimation process.

Recently, the information form of SLAM has become very popular. Filter-based approaches include the sparse extended information filter (SEIF) by Thrun *et al.* [133] and the thin junction tree filter (TJTF) by Paskin [115]. On the smoothing side, Treemap by Frese [52] exploits the information form, but applies multiple approximations to provide a highly efficient algorithm. Square root SAM by Dellaert and Kaess [29, 32] provides an efficient and exact solution based on a batch factorization of the information matrix, but does not address how to efficiently access the marginal covariances.

While SAM includes the complete trajectory and map, this is not always the case when smoothing is applied. Instead, the complexity of the estimation problem can be reduced by omitting the trajectory altogether, as for example pursued by Wang *et al.* [137] with D-SLAM, where measurements are transformed into relative constraints between landmarks. Similarly, parts of the trajectory can be omitted as done by Folkesson and Christensen [46], where parts of the underlying graph are collapsed into so-called star nodes. Alternatively, the problem can be stated as estimating the trajectory only, which leads to an exactly sparse information matrix by Eustice *et al.* [41, 44], where image measurements are converted to relative pose constraints. While this approach is similar to my pose-only case, they employ iterative methods to solve the estimation problem. While conservative covariance estimates are available in Eustice's work [41], and in fact were the inspiration for my work, efficient access to the exact covariances is not possible based on the iterative solver.

A completely different approach to SLAM that keeps the complete trajectory is FastSLAM by Montemerlo *et al.* [99, 100, 69, 63], which employs a Rao-Blackwellized particle filter that keeps multiple hypotheses of robot trajectories, each with its own map. While this theoretically keeps a complete posterior over the robot trajectory, one problem in practice is particle depletion, as the dimension of the state increases, while the number of particles stays constant. Another approach to SLAM is to split the estimation problem into separate mapping and localization problems, which are then iteratively solved in an expectation maximization (EM) framework [132, 3]. For large-scale environments, the SLAM problem can also be split into several smaller, connected mapping problems, or submaps, as done in the Atlas approach by Bosse *et al.* [10, 11].

Recently, some SLAM algorithms employ direct equation solvers based on Cholesky or QR factorization. Treemap [54] uses Cholesky factors to represent probability distributions in a tree-based algorithm. However, in contrast to SAM, approximations are employed in order to reduce the computational complexity.

Chapter III

ISAM: INCREMENTAL SMOOTHING AND MAPPING

In this chapter I present incremental smoothing and mapping (iSAM), which performs *fast incremental updates* of the square root information matrix yet is able to compute the full map and trajectory at any time. Our previous work, SAM, is a batch algorithm that first updates the information matrix when new measurements become available and then factors this new information matrix. Hence, it performs unnecessary calculations when applied incrementally as it recalculates all entries, even the ones that do not change. In contrast, for iSAM I directly update the square root information matrix with new measurements as they arrive using standard matrix update equations [60]. That means I reuse the previously calculated components of the square root factor and perform calculations only for entries that are actually affected by the new measurements. Thus, I obtain a local and constant time operation for exploration tasks, that is for trajectories without loops for which the robot does not revisit previously mapped locations.

For trajectories with loops, *periodic variable reordering* prevents unnecessary fill-in in the square root factor that would otherwise slow down the incremental factor update as well as the recovery of the current state estimate by back-substitution. Fill-in is a well-known problem for sparse matrix factorization, as the resulting matrix factor can contain a large number of additional non-zero entries that reduce or even destroy the sparsity with associated negative consequences for the computational complexity. As the variable ordering influences the amount of fill-in obtained, it allows us to influence the computational complexity involved in solving the system. While finding the variable ordering that achieves the lowest fill-in is infeasible, good heuristics are available. I perform incremental updates most of the time but periodically apply a variable reordering heuristic, followed by refactoring the resulting measurement Jacobian that can optionally also contain relinearized measurements.

It should be noted that iSAM is a general estimation method that can be applied to any

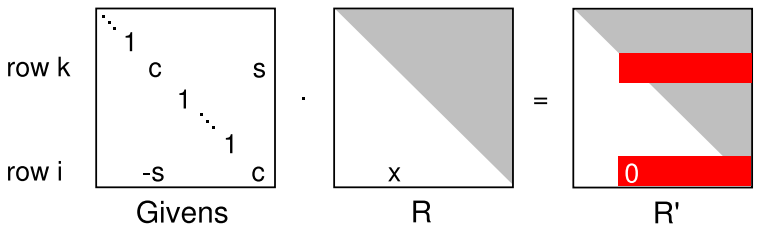


Figure 12: Using a Givens rotation as a step in transforming a general matrix into upper triangular form. The entry marked "x" is eliminated, changing some of the entries marked in red (dark), depending on sparsity.

constraints between variables and does not require the presence of landmarks. In particular, I present a pose-only variant of iSAM, where instead of landmark measurements arbitrary pose constraints are used, for example derived from dense laser scan matching.

I evaluate iSAM on simulated and real-world datasets for both landmark-based and pose-only settings. The results show that iSAM provides an efficient and exact solution for both types of SLAM settings. They also show that the square root factor indeed remains sparse even for large-scale environments with a significant number of loops.

The following sections begin with a review of Givens rotations for batch and incremental QR matrix factorization. I then apply this technique to updating of the square root factor and discuss how to retrieve the map and trajectory. I continue by discussing how to use periodic variable reordering to deal with loops and how to deal with nonlinear measurements. I provide experimental results from iSAM applied in different contexts. I finally discuss the computational complexity of iSAM for different scenarios.

3.1 Matrix Factorization by Givens Rotations

A standard approach to obtain the QR factorization of a matrix A uses *Givens rotations* [60] to zero out all entries below the diagonal, one at a time. While this is not the preferred way to do full QR factorization, we will later see that this approach readily extends to factorization updates, which are needed to incorporate new measurements. The process starts from the left-most non-zero entry, and proceeds column-wise, by applying the Givens

rotation

$$\Phi := \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \tag{19}$$

to rows i and k, with $i > k$ as shown in Figure 12. The parameter ϕ is chosen so that a_{ik}, the (i,k) entry of A, becomes 0. After all entries below the diagonal are zeroed out in this manner, the upper triangular entries contain the square root factor R. The orthogonal rotation matrix Q is typically dense, which is why this matrix is never explicitly formed in practice. Instead, it is sufficient to update the right-hand side (RHS) vector $\mathbf{b}$ with the same rotations that are applied to A.

Solving a least squares system $A\mathbf{x} = \mathbf{b}$ by matrix factorization using Givens rotations is numerically stable and accurate to machine precision if the rotations are determined as follows [60, Sec. 5.1]:

$$(\cos\phi, \sin\phi) = \begin{cases} (1,0) & \text{if } \beta = 0 \\ \left(\frac{-\alpha}{\beta\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}}, \frac{1}{\sqrt{1+\left(\frac{\alpha}{\beta}\right)^2}} \right) & \text{if } |\beta| > |\alpha| \\ \left(\frac{1}{\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}}, \frac{-\beta}{\alpha\sqrt{1+\left(\frac{\beta}{\alpha}\right)^2}} \right) & \text{otherwise} \end{cases} \tag{20}$$

where $\alpha := a_{kk}$ and $\beta := a_{ik}$.

3.2 Incremental Factorization Updates for SAM

When a new measurement arrives, it is more efficient to modify the previous factorization directly by *QR-updating*, instead of updating and refactoring the measurement Jacobian A. Adding a new measurement row $\mathbf{w}^T$ and RHS γ into the current factor R and RHS $\mathbf{d}$ yields a new system that is not yet in the correct factorized form:

$$\begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} \begin{bmatrix} A \\ \mathbf{w}^T \end{bmatrix} = \begin{bmatrix} R \\ \mathbf{w}^T \end{bmatrix}, \text{ new RHS: } \begin{bmatrix} \mathbf{d} \\ \gamma \end{bmatrix}. \tag{21}$$

Note that this is the same system that is obtained by applying Givens rotations to the updated matrix A' to eliminate all entries below the diagonal, except for the last (new) row. Therefore Givens rotations can be determined that zero out this new row, yielding the updated factor R'. In the same way as for the full factorization, we simultaneously update

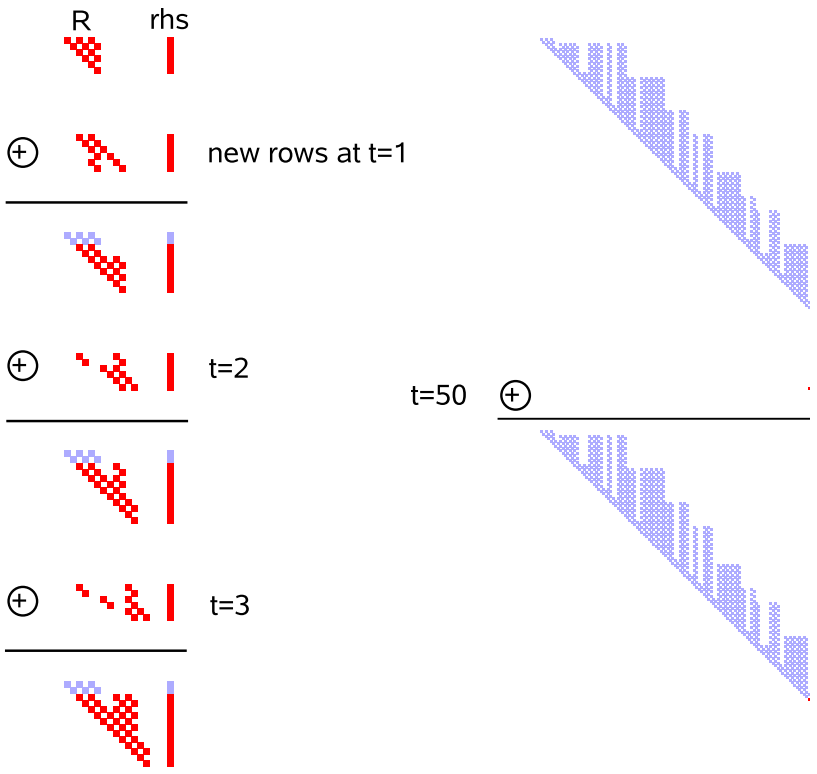


Figure 13: Updating the square root factor for the example of an exploration task: New measurement rows are added to the upper triangular factor R and the right-hand side (RHS). The left column shows the updates for the first three steps, and the right column shows the update after 50 steps. The update operation is symbolically denoted by $\oplus$. Entries that remain unchanged are shown in light blue (gray). Note that for an exploration task, the number of operations is bounded by a constant.

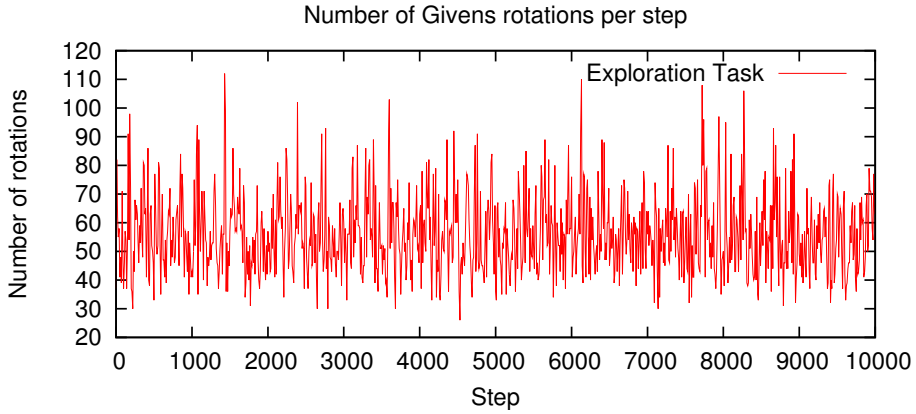


Figure 14: Number of Givens rotations needed per step for a simulated linear exploration task. In each step, the square root factor is updated by adding the new measurement rows using Givens rotations. The number of rotations is independent of the length of the trajectory.

the RHS with the same rotations to obtain $\mathbf{d}'$. Several steps of this update process are shown in Figure 13.

New variables are added to the QR factorization by expanding the factor R by the appropriate number of empty columns and rows. This expansion is simply done before new measurement rows containing the new variables are added. At the same time, the RHS $\mathbf{d}$ is augmented by the same number of zeros.

Applying the Givens-rotations-based updating process to the square root factor provides the basis for my efficient incremental solution to smoothing and mapping. In general, the maximum number of Givens rotations needed for adding a new measurement row is n. However, as both R and the new measurement row are sparse, only a constant number of Givens rotations are needed. Furthermore, new measurements typically refer to recently added variables, so that often only the rightmost part of the new measurement row is (sparsely) populated.

For a linear exploration task, incorporating a set of new landmark and odometry measurements takes constant time. Exploration means that the robot does not revisit previously mapped areas, or at least that such a revisiting is not identified. Examples of such updates are shown in Figure 13. The simulation results in Figure 14 show that the number of rotations needed is independent of the size of the trajectory and the map. Updating the square

root factor therefore takes $O(1)$ time, but this does not yet provide the current least squares estimate.

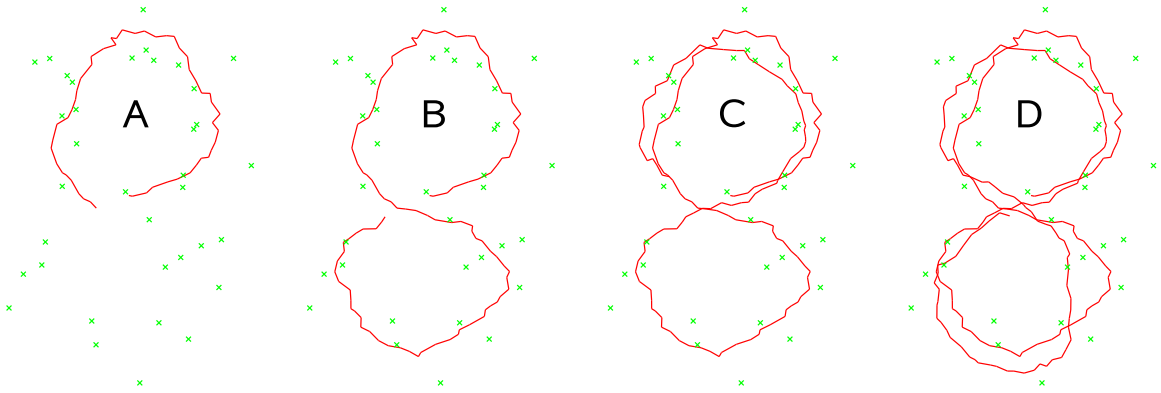
The current least squares estimate for the map and the full trajectory can be obtained at any time by back-substitution in time linear in the number of variables. While back-substitution has quadratic time complexity for general dense matrices, it is more efficient in the context of iSAM. For exploration tasks, the information matrix is band-diagonal. Therefore, the square root factor has a constant number of entries per column independent of the number of variables n that make up the map and trajectory. Therefore, back-substitution requires $O(n)$ time in iSAM. In the linear exploration example from above, this results in about $0.12s$ computation time after 10 000 steps.

3.3 Loops and Periodic Variable Reordering

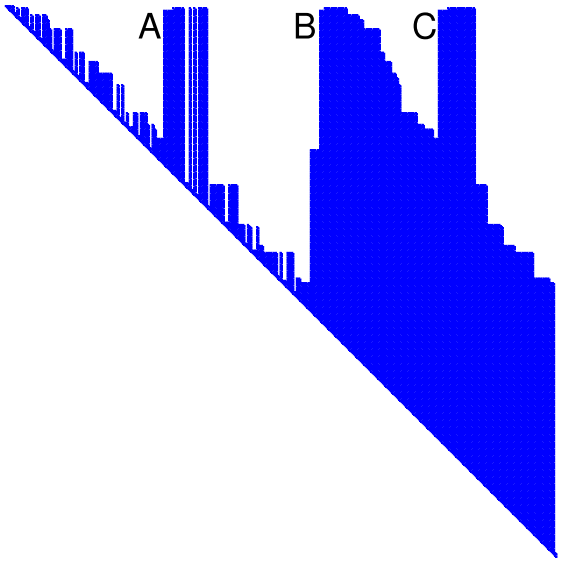
I discuss how iSAM keeps the computational requirements low even if there are loops in the trajectory. While realistic SLAM applications include much exploration, the robot often returns to previously visited places, closing loops in the trajectory. I discuss the consequences of loops on the matrix factorization and show how to use periodic variable reordering to avoid unnecessary increases in computational complexity.

Environments with loops do not have the nice property of local updates, resulting in increased complexity. In contrast to pure exploration, where a landmark is only visible from a short local part of the trajectory, a loop brings the observer back to a previously visited location. A loop introduces correlations between the current pose and previously observed landmarks, which themselves are connected to earlier parts of the trajectory. An example based on a simulated environment with a robot trajectory in the form of a double figure eight loop is shown in Figure 15.

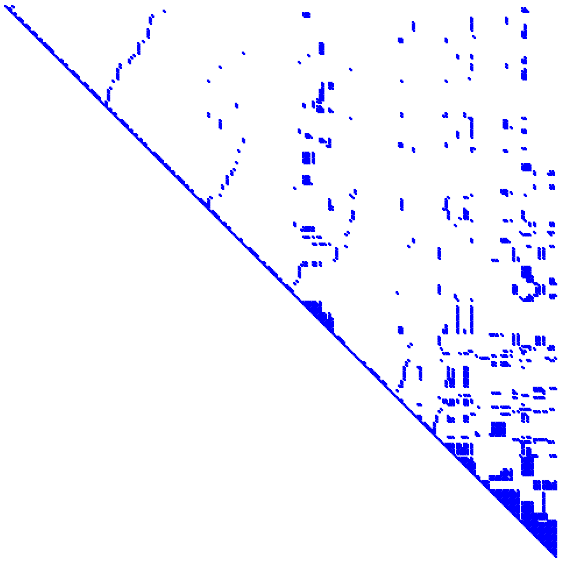
Loops in the trajectory can result in a significant increase in computational complexity through a large increase of non-zero entries in the factor matrix. Non-zero entries beyond the sparsity pattern of the information matrix are called *fill-in*. While the smoothing information matrix remains sparse even in the case of closing loops, the incremental updating of the factor matrix R leads to fill-in as shown in Figure 15(b).



(a) Simulated double figure eight loop at interesting stages of loop closing.



(b) Triangular factor matrix R.



(c) The same factor R after variable reordering.

Figure 15: For a simulated environment consisting of a figure eight loop that is traversed twice (a), the upper triangular factor R shows significant fill-in (b). Some fill-in occurs at the time of the first loop closing (A). Note that no more fill-in occurs on the subsequent exploration along the second loop until the next loop closure occurs (B). However, the fill-in then becomes significant when the complete figure eight loop is traversed for the second time, with a peak when visiting the center point of the figure eight loop for the third time (C). After variable reordering according to an approximate minimum degree heuristic, the factor matrix again is completely sparse (c).

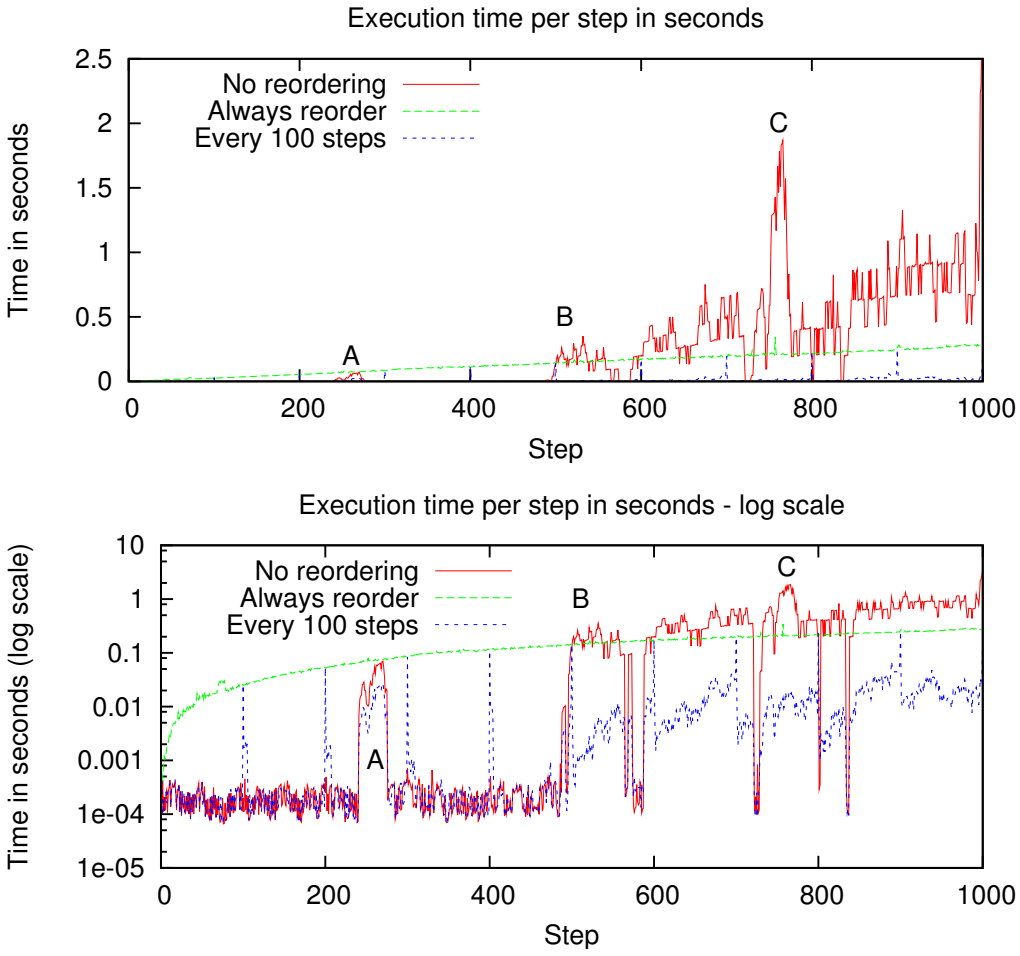


Figure 16: Comparison of execution times for the examples in Figure 15, shown in both linear (top) and log scale (bottom). Adding variables in the order that they appear in the estimation problem yields bad performance (continuous red) due to significant matrix fill-in as shown in Figure 15(b). Even reordering the variables after each step (dashed green) is sometimes less expensive. However, a considerable increase in efficiency is achieved by using fast incremental updates interleaved with only occasional variable reordering (dotted blue), here performed every 100 steps. Note especially that in the linear time figure this curve is near zero except for spikes caused by the periodic reordering.

I avoid fill-in by *variable reordering*, a technique well known in the linear algebra community. The order of the columns in the information matrix influences the variable elimination order and therefore also the resulting number of entries in the factor R. While obtaining the best column variable ordering is NP hard, efficient heuristics such as the COLAMD (*col*umn *a*pproximate *m*inimum *d*egree) ordering by Davis *et al.* [19] have been developed that yield good results for the SLAM problem as shown in [29, 32]. I apply this ordering heuristic to blocks of variables that correspond to the robot poses and landmark locations. As has been shown in [32], operating on these blocks leads to a further increase in efficiency as it exploits the special structure of the SLAM problem. The corresponding factor R after applying the COLAMD ordering shows negligible fill-in, as shown in Figure 15(c).

I propose fast incremental updates with periodic variable reordering, combining the advantages of both methods. Factorization of the new measurement Jacobian after variable reordering is expensive when performed in each step. But combined with incremental updates it avoids fill-in and still yields a fast algorithm as supported by the timing results in Figure 16. In fact, as the logarithmic scale version of this figure shows, my solution is one to three orders of magnitude faster than either the purely incremental or the batch solution with the exception of occasional peaks caused by deploying the batch solution for variable reordering and subsequent matrix factorization. In this example I use a fixed interval of 100 steps after which I reorder the variables and refactor the complete matrix.

3.4 Nonlinear Functions

I discuss how relinearization allows iSAM to still provide an exact solution even for nonlinear measurement functions. While I have so far only discussed the case of linear measurement functions, SLAM applications usually are faced with nonlinear measurement functions. Angular measurements, such as the robot orientation or the bearing to a landmark, are the main source for nonlinear dependencies between measurements and variables. In the nonlinear case everything discussed so far is still valid. However, after solving the linearized version based on the current variable estimate we might obtain a better estimate resulting in a modified measurement Jacobian A based on this new linearization point. For standard

nonlinear optimization techniques this process is iterated as explained in Section 2.3 until the change in the estimate is sufficiently small. Convergence is guaranteed, at least to a local minimum, and the convergence speed is quadratic because my QR factorization based direct solver represents a second order method.

As relinearization is not needed in every step, I propose combining it with periodic variable reordering. The SLAM problem is different from a standard nonlinear optimization as new measurements arrive sequentially. First, we already have a very good estimate for most if not all of the old variables. Second, measurements are typically fairly accurate on a local scale, so that good estimates are also available for the newly added robot pose as well as for newly added landmarks. We can therefore avoid calculating a new Jacobian in each step and refactoring it, a potentially expensive batch operation. Instead, for the results presented here, I combine relinearization with the periodic variable reordering used for fill-in reduction as discussed in the previous section. In other words, in the variable reordering steps only, I also relinearize the measurement functions as the new measurement Jacobian has to be refactored anyways.

3.5 Experiments and Results

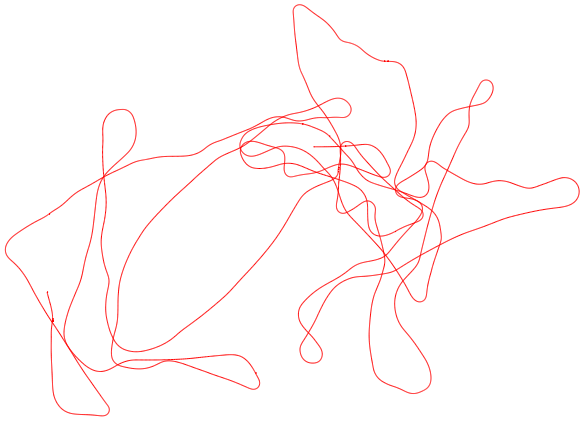
I evaluate the overall iSAM algorithm on simulated data as well as real-world datasets. The simulated data allow comparison with ground-truth, while the real-world data prove the applicability of iSAM to practical problems. I explore both landmark-based as well as pose-constraint-based SLAM.

I have implemented iSAM in the functional programming language OCaml, using exact, automatic differentiation [62] to obtain the Jacobians. All timing results in this section are obtained on a Core 2 Duo 2.2 GHz laptop computer, using only a single core.

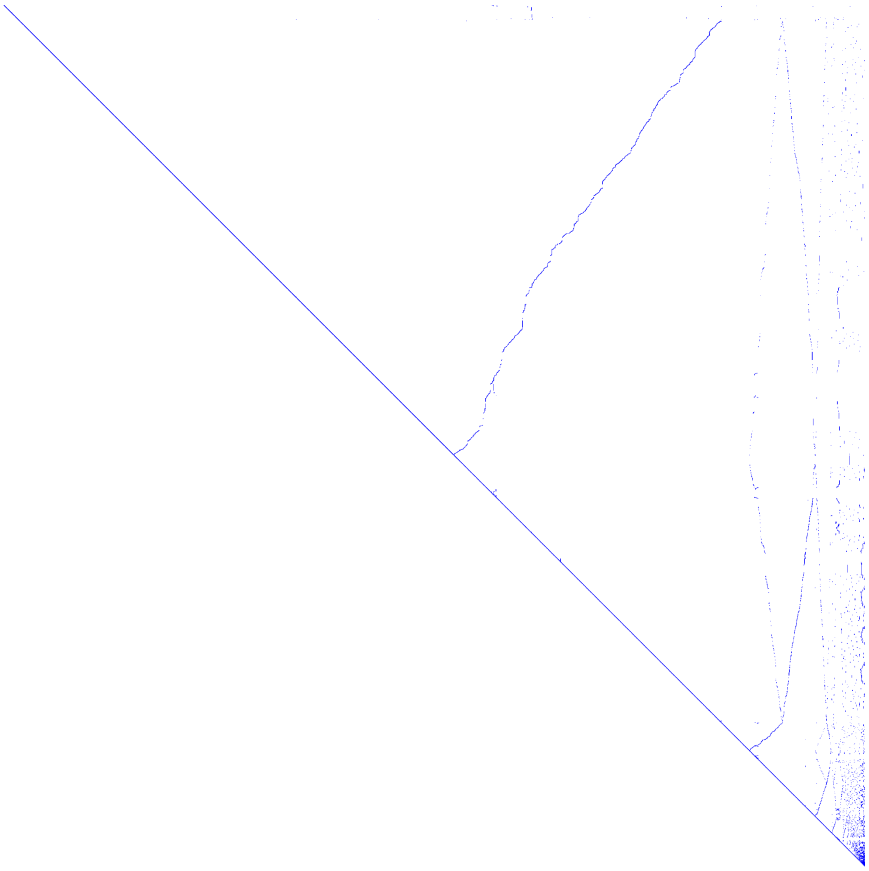
3.5.1 Landmark-based iSAM

I show that landmark-based iSAM works well in real-world settings, even in the presence of many loops in the robot trajectory. I evaluate iSAM on the Sydney Victoria Park dataset (see Figure 17), a popular test dataset in the SLAM community that consists of laser-range data and vehicle odometry, recorded in a park with sparse tree coverage. It contains 7247

Figure 17: Results for the full Victoria Park sequence. Solving the complete problem including data association in each step took 3.4 minutes on a laptop computer. Since the dataset is from a 26 minute long robot run, iSAM is over **7 times faster than real-time** in this case, calculating the complete and exact solution for each of the almost 7000 steps. The trajectory and landmarks are shown in yellow (light), manually overlaid on an aerial image for reference. Differential GPS was not used in obtaining my experimental results, but is shown in blue (dark) for comparison - note that in many places GPS was not available, presumably due to obstruction by trees.



(a) Trajectory based on odometry only.



(b) Final matrix factor R with side length 21 187.

Figure 18: The vehicle trajectory according to odometry (a) and the final sparse square root factor (b) for the Victoria Park dataset shown in Figure 17. Note that the final R factor with side length over 21 000 is indeed sparse with approximately 10 entries per column, taking about $1MB$ of memory including book keeping, while a dense matrix of this size would require $1712MB$ of memory.

frames along a trajectory of 4 kilometer length, recorded over a time frame of 26 minutes. As repeated measurements taken by a stopped vehicle do not add any new information, we have dropped these, leaving 6969 frames. We have extracted 3640 measurements of landmarks from the laser data by a simple tree detector. The robot trajectory derived from vehicle odometry is shown in Figure 18(a).

iSAM with known correspondences runs comfortably in real-time; results for unknown data association are provided in Chapter 4. The incremental reconstruction including *solving for all variables for each new frame that is added* took $201s$ or 3.4 minutes, which is significantly less than the 26 minutes it took to record the data. That means that even though the Victoria Park trajectory contains a significant number of loops (several places are traversed 8 times), increasing fill-in, iSAM is still over 7 times faster than real-time. The resulting map contains 140 distinct landmarks as shown in Figure 17. Solving after every step is in fact not necessary, as the measurements are fairly accurate locally, thereby providing good estimates. Obtaining the exact map and trajectory by back-substitution only every 10 steps yields a significant improvement to $93s$ or 1.6 minutes.

More importantly, iSAM still performs in real-time towards the end of the trajectory, where the computations get more expensive. The average calculation time for the final 100 steps are the most expensive ones to compute as the number of variables is largest. Even for the slow case of retrieving the full solution after every step, iSAM takes on average $50ms$ per step for the last 100 steps. This result compares favorably to the $220ms$ needed for real-time performance. These computation times include a full linearization, COLAMD variable reordering step and matrix factorization, which took $1.1s$ in total. Despite all the loops, the final factor R as shown in Figure 18(b) is still sparse, with 207 422 entries for 21 187 variables, yielding an average of only 9.79 entries per column.

3.5.2 Pose-Constraint-based iSAM

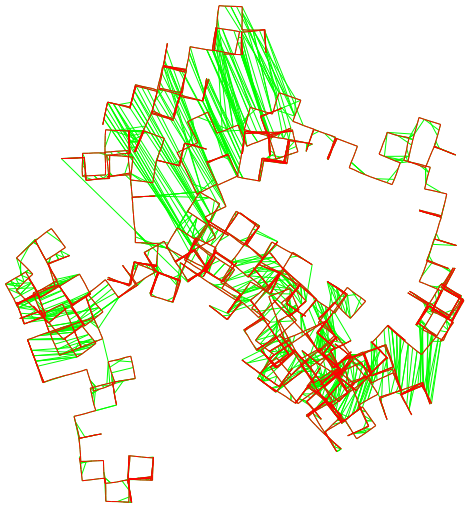
iSAM can straightforwardly be applied to estimation problems without landmarks, purely based on pose constraints. Such pose constraints most commonly arise from scan-matching dense laser range data, but can also be generated from visual input [42, 87]. Pose constraints

either connect subsequent poses similar to odometry measurements, or they connect two arbitrary poses when closing loops. Pose constraints are implemented in iSAM as terms that represent the error between the predicted and the measured difference between a pair of poses, much in the same way as odometry measurements, but without the requirement that the pair of poses has to be two subsequent poses in the trajectory. I evaluate pose-only iSAM on simulated as well as real-world datasets, assuming known data association, as the generation of pose constraints by scan-matching has been well studied and good algorithms are available [64, 69].

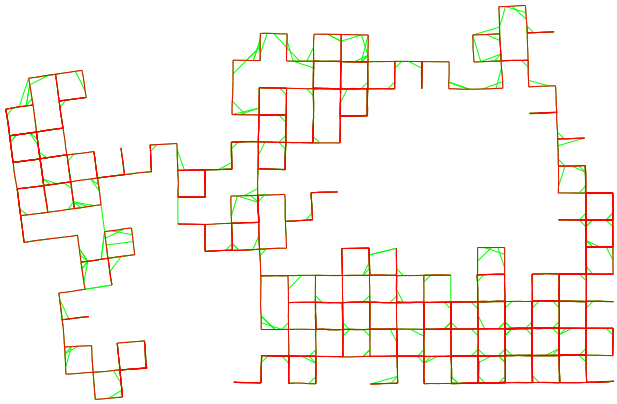
3.5.2.1 Simulated Data

The incremental solution of iSAM is comparable in quality to the solution obtained by full nonlinear optimization. To allow ground truth comparison, I use the *simulated Manhattan world* from [113] shown in Figure 19(a),(b). This dataset contains 3500 poses and 5598 constraints, 3499 of which are odometry measurements. While the result in [113] seems to be better as the left part is more straightened out, my solution has a slightly lower normalized χ^2 value of 1.0406, compared to 1.0412. After one extra relinearization and back-substitution, the normalized χ^2 is 1.0375, the same value that we obtain by full nonlinear optimization until convergence. In Figure 20 the normalized χ^2 value for every step of the incremental solution is compared with the batch SAM solution that iterates until convergence. Short spikes are caused by temporary linearization errors when loops are closed and disappear during relinearization, which is performed in combination with the batch variable reordering steps. Except for these spikes iSAM performs like a full direct (second-order) equation solver, while other incremental (first-order) equation solvers, such as [114] always provide a slightly increased normalized χ^2 value. Monitoring the χ^2 value could provide a criterion for when to perform relinearization, although I have not done so yet. These results show that iSAM is generally comparable in accuracy to the exact solution provided by the batch SAM algorithm.

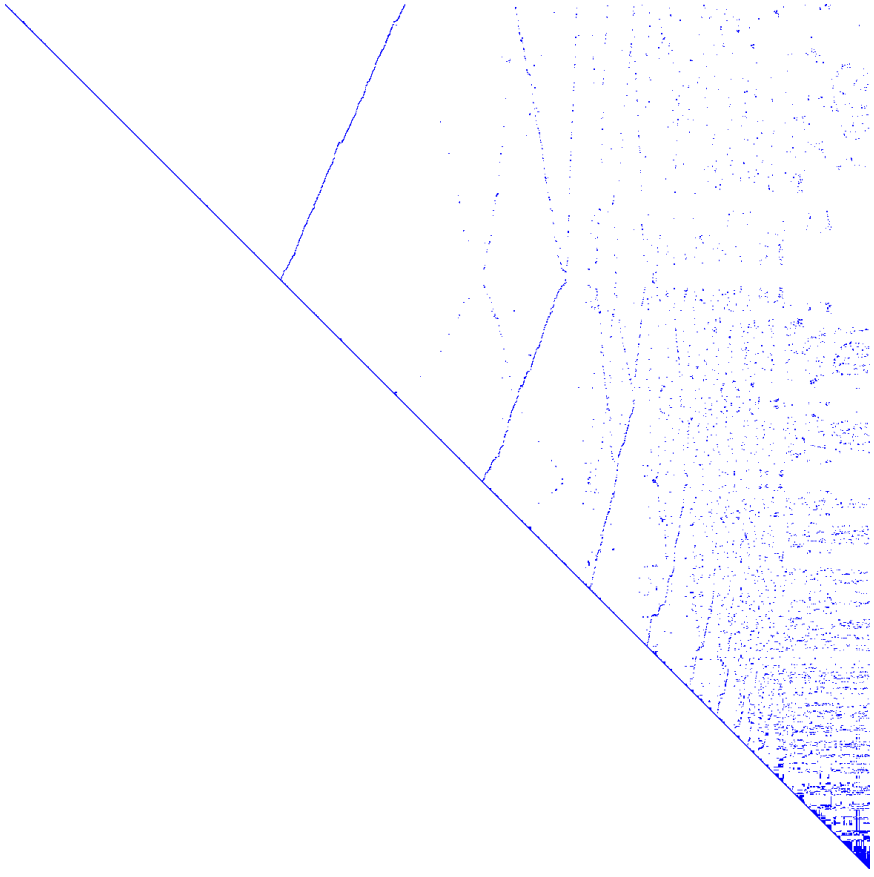
In terms of computational speed, iSAM also fares well for this dataset. Solving the full problem for each newly added pose, while reordering the variables and relinearizing the



(a) Original noisy data set.



(b) Trajectory after incremental optimization.



(c) Final factor matrix R with side length 10 500.

Figure 19: iSAM results for the simulated Manhattan world from [113] with 3500 poses and 5598 constraints. iSAM takes about $24ms$ per step. The resulting R factor has 187 423 entries, which corresponds to 0.34% or an average of 17.8 entries per column.

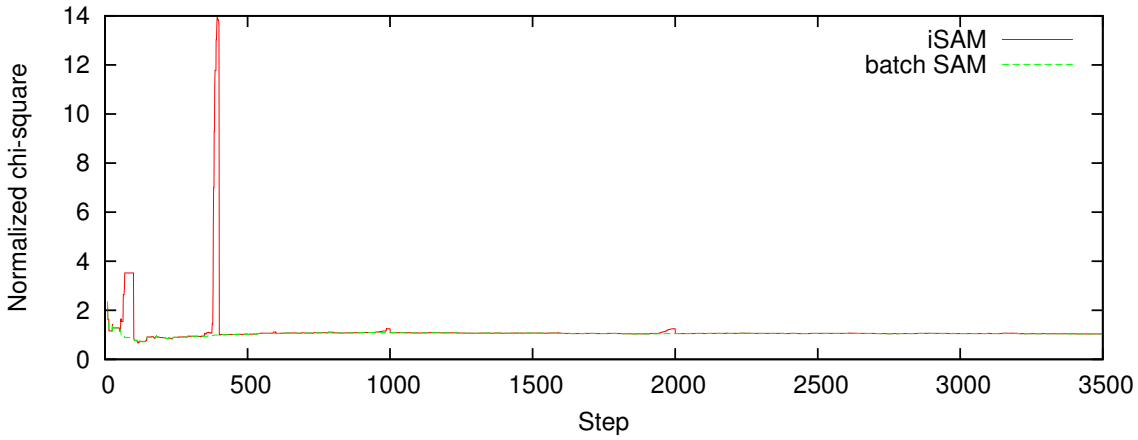
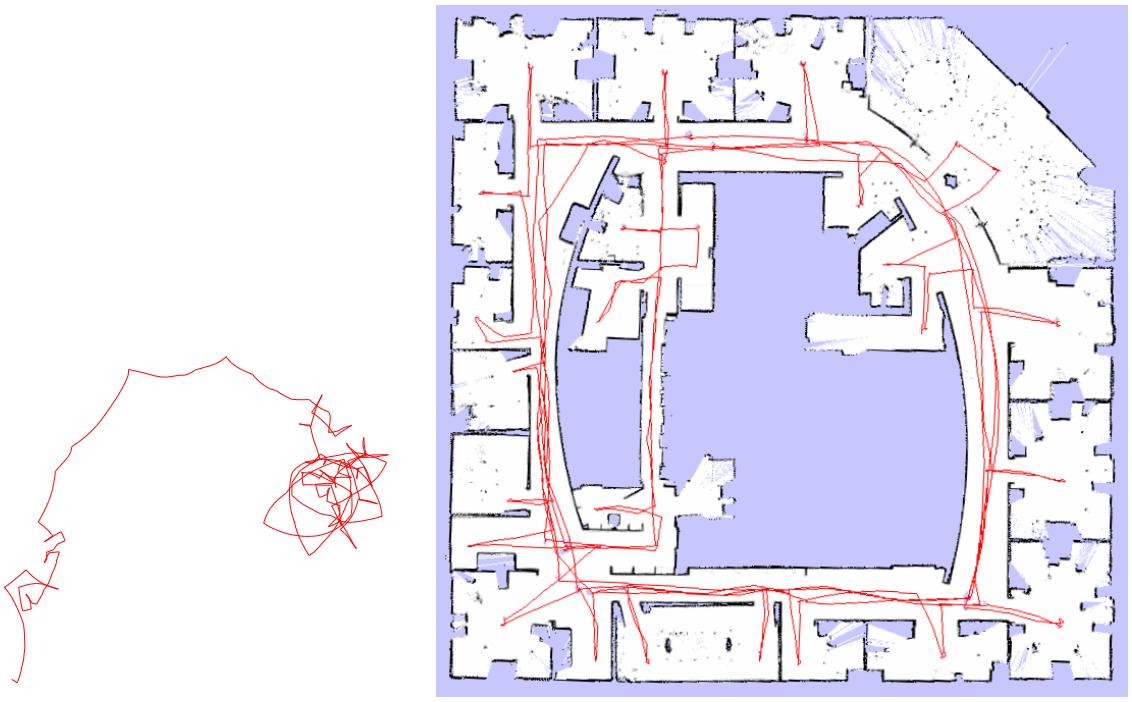


Figure 20: iSAM generally yields an exact solution, as shown by the normalized chi-square values for each iteration on the Manhattan world from Figure 19. I compare iSAM with a full nonlinear batch solution that iterates until convergence. The spikes are caused by temporary linearization errors after loop closing and always vanish during the periodic variable reordering step that also performs relinearization – see for example steps 400 and 1000.

problem every 100 steps, takes iSAM 83.0s, or an average of 23.7ms per step. The last 100 steps take an average of 30ms each, which includes 0.76s for variable reordering and matrix factorization. The resulting R factor shown in Figure 19(c) is sparse with 187 423 entries for a side length of 10 500.

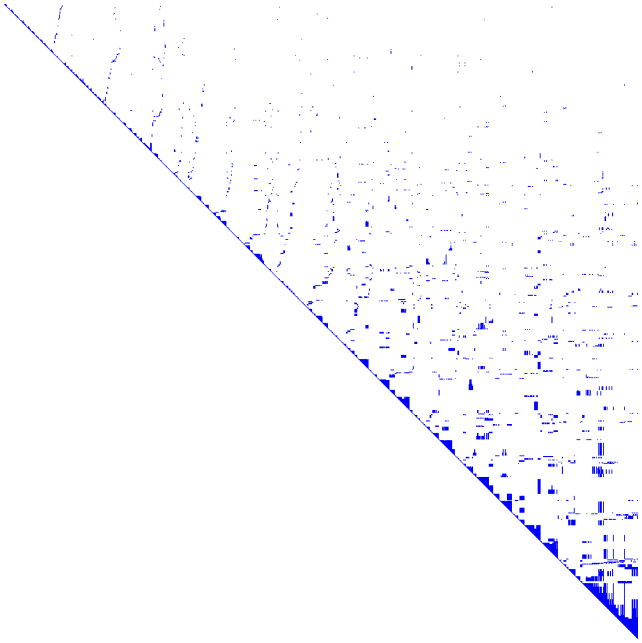
3.5.2.2 Real-world data

iSAM also performs well on real-world data, both in quality of the solution as well as speed. I apply iSAM to two publicly available laser range datasets that appear in several publications. The first one is the *Intel dataset* shown in Figure 21(b), providing a trajectory with many loops with continued exploration of the same environment in increasing detail. Preprocessing by scan matching results in 910 poses and 4453 constraints. iSAM obtains the full solution after each step with variable reordering every 20 frames in 44.9s, or about 49ms per step. Our results are of the same order as the ones presented in state of the art work by Olson *et al.* [114] (45s using a Java implementation on a 2.4GHz desktop, but also using 1500 more constraints due to different preprocessing). The last 100 steps take an average of 111ms including 0.51s for each of the five reordering and factorization steps. The final R factor is shown in Figure 21(c), with 2730 variables containing 90 363 entries.



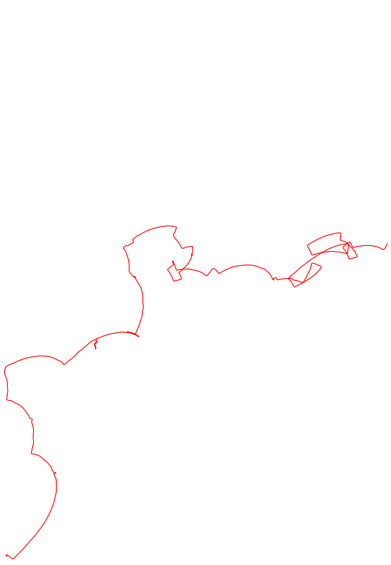
(a) Trajectory based on odometry.

(b) Final trajectory and evidence grid map.

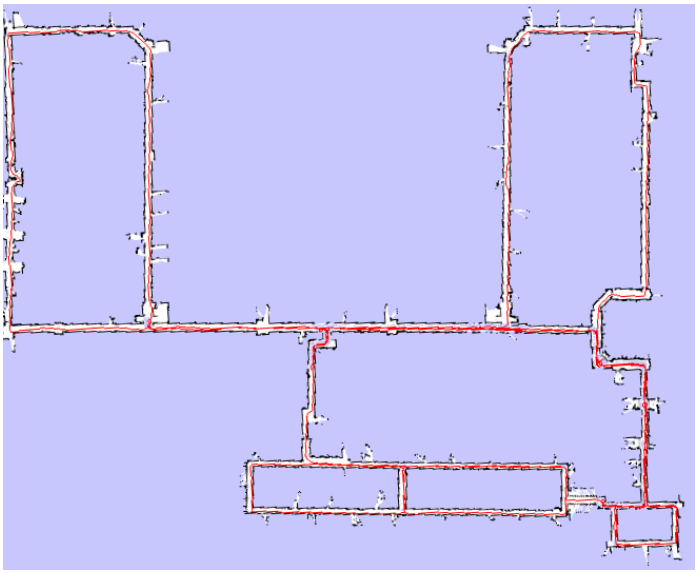


(c) Final factor matrix R with side length 2730.

Figure 21: Results from iSAM applied to the Intel dataset. iSAM calculates the full solution for 910 poses and 4453 constraints with an average of $49ms$ per step, while reordering the variables every 20 steps. The problem has $910 \times 3 = 2730$ variables and $4453 \times 3 = 13\,359$ measurement equations. The R factor contains 90 363 entries, which corresponds to 2.42% or 33.1 entries per column.



(a) Trajectory based on odometry.



(b) Final trajectory and evidence grid map.

(c) Final factor matrix R with side length 5823.

Figure 22: iSAM results for the MIT Killian Court dataset. iSAM calculates the full solution for the 1941 poses and 2190 pose constraints with an average of 7.3ms per step. The R factor contains 52 414 entries for 5823 variables, which corresponds to 0.31% or 9.0 per column.

The second real-world pose-only example I use to evaluate iSAM is the *MIT Killian Court dataset* shown in Figure 22(b) that features much exploration with a few large-scale loops. Again the dataset was preprocessed, yielding 1941 poses and 2190 pose constraints. iSAM takes $14.2s$ for a complete solution after each step, or about $7.3ms$ per step, with variable reordering after every 100 steps. The last 100 steps take an average of $18ms$ including $0.25s$ for reordering/refactorization. The final R factor with 52 414 entries is shown in Figure 22(c).

3.5.3 Sparsity of the Square Root Factor

The complexity of iSAM heavily depends on the sparsity of the square root factor, as this affects both retrieving the solution as well as access to the covariances as discussed in the next chapter. Retrieving the solution from the square root factor requires back-substitution, which usually has quadratic time complexity. However, if there are only a constant number of entries per column in the square root factor as is true for the case of exploration tasks, then back-substitution only requires $O(n)$ time.

My results show that the number of entries per column is typically bound by a low constant. Figure 23 shows how the density of the factor matrix R develops over time for each dataset used in this chapter. The densities initially increase, showing very large changes: Increases are caused by incremental updating of the matrix factor, while sudden drops are the consequence of the periodic variable reordering. Most curves clearly converge to a low constant, explaining the good performance of iSAM. For the Intel dataset and the Manhattan world, the density increases more significantly because the number of constraints per step, that is the density of measurements, increases. The Intel sequence contains fairly dense constraints, which is also the reason for choosing a shorter interval for the periodic variable reordering (20 steps) than for all other datasets (100 steps). Nevertheless, as the results for that sequence show, iSAM still performs many times faster than needed for real-time.

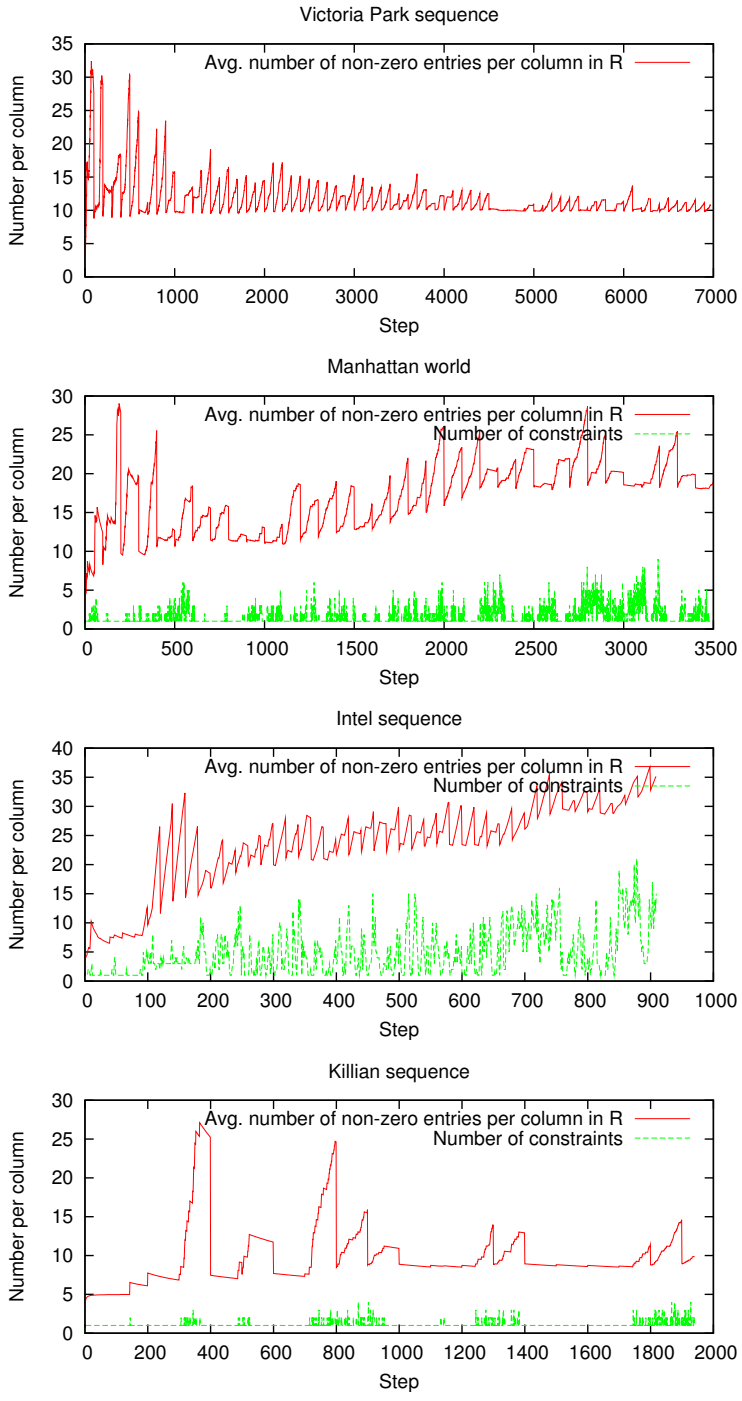


Figure 23: Average number of entries per column in the R factor over time for the different datasets in this section. Even though the environments contain many loops, the average converges to a low constant or is at least proportional to the number of constraints per step. This confirms my assumption that the number of entries per column is approximately independent of the number of variables n.

Table 1: Computational complexity of iSAM under various assumptions on the sparsity of the R factor. Note that $n = Md_{\mathbf{x}} + Nd_{\mathbf{l}}$ depends on both the number of landmarks N as well as the number of poses M.

	update	back-substitution	batch factorization
R banddiagonal (exploration task)	$O(1)$	$O(n)$	$O(n)$
dense blocks in R, but N constant (remaining in the same room)	$O(1)$	$O(n)$	$O(n)$
$\log n$ entries per column avg. (planar graph, see text)	$\geq O(\log n)$ $\leq O(n)$	$O(n \log n)$	$O(n^{1.5})$
dense R (not encountered so far)	$O(n)$	$O(n^2)$	$O(n^3)$

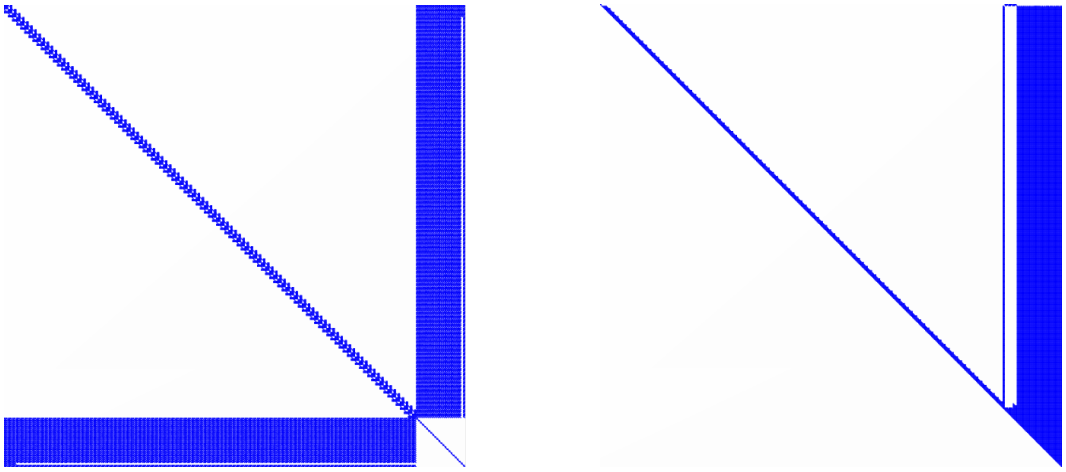
3.6 Computational Complexity

In this section I analyze the computational complexity of iSAM under various assumptions on the nature of the mapping task. While it is difficult to make general statements about the computational complexity, theoretical bounds are available when one makes certain assumptions about the sparsity of the factor matrix. These assumptions again are connected to certain types of environments and mapping tasks as discussed in this section. The results are summarized in Table 1. As before, I use n as the number of variables in the system, which is equal to the side length of the R factor. Note that $n = Md_{\mathbf{x}} + Nd_{\mathbf{l}}$ depends on both the number of poses M and the number of landmarks N, where $d_{\mathbf{x}}$ and $d_{\mathbf{l}}$ are the dimensions of individual pose and landmark variables, respectively.

3.6.1 Exploration Task

The simplest case to discuss is a pure *exploration* task, in which the robot never returns to previously mapped places, or at least does not identify previously mapped places and therefore never closes any loops. For that case the information matrix is band-diagonal, and therefore also the factor matrix under a suitable variable ordering. For a band-diagonal factor matrix, the fill-in per column is bounded by a constant or $O(1)$.

Updating the square root factor R with new measurement rows takes constant time for exploration. We have to apply Givens rotations to each entry of the new measurement



(a) Information Matrix (b) Factor matrix R

Figure 24: When all landmarks are visible at all times we obtain dense blocks in the information matrix as well as in the factor matrix.

rows, and we know that there are a constant number of measurements per row. However, these introduce new non-zero entries depending on the fill-in of the factor matrix R, which in our case is constant, that again require Givens rotations to zero them out. Each Givens rotation takes constant time, resulting in $O(1)$ time for updating the factor matrix.

Recovering the solution by back-substitution for both trajectory and map takes linear time. In general, back-substitution is a quadratic operation, but for our sparse matrix it takes $O(n)$ time, as n elements need to be calculated, and for each we have to substitute a constant number of entries according to one row of the factor matrix.

Batch solution of the complete system also takes linear time. It is well known that factorization of a band-diagonal matrix takes $O(n)$ computation time. Additionally, back-substitution for recovering the solution again takes $O(n)$ time.

3.6.2 Remaining in the Same Room

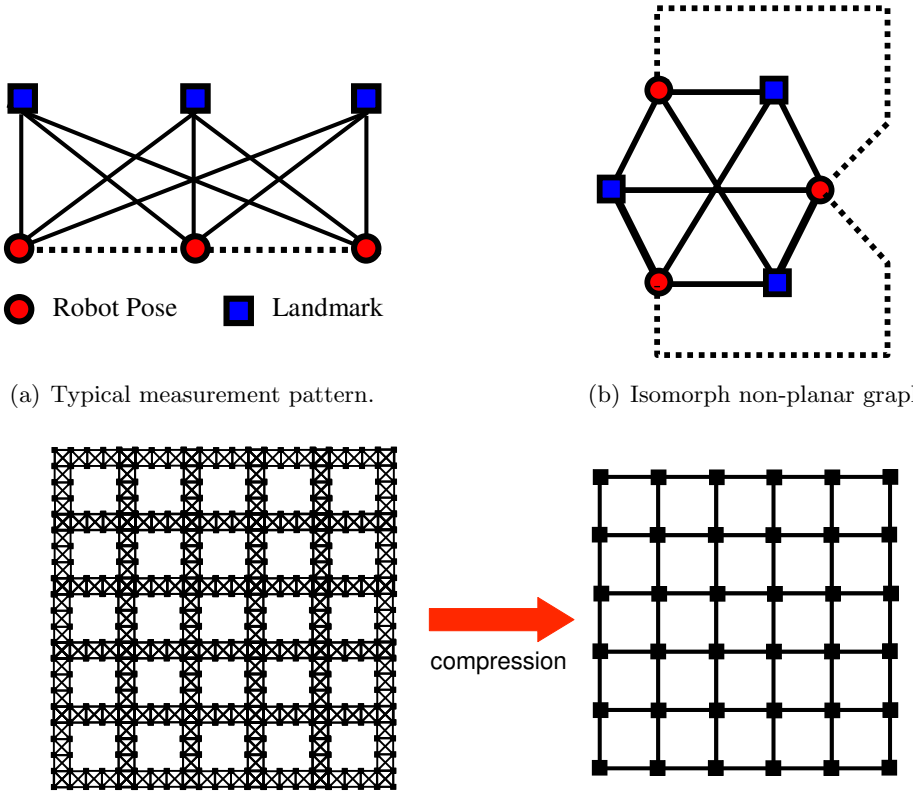
The opposite of an exploration task is a robot that continuously stays in the same environment and always observes the same landmarks, thereby continuously closing loops. While this at first seems to be a worst case scenario for iSAM, the computational complexity is actually equal to the exploration case, but with a potentially large constant added. The key insight is that the number of landmarks are constant in this case. Figure 24 shows the

factor matrix R as well as the information matrix $R^T R$ for a simulated environment where the robot is nearly stationary and continuously observes the same landmarks. The simulation contains 101 poses with odometry measurements, 24 landmarks and 2326 landmark measurements. The factor matrix R shown in Figure 24 can clearly not be considered to be sparse anymore, with about 21% non-zero entries. The information matrix itself contains dense blocks for this special case, and we cannot expect to obtain a more sparse factor matrix. Note that the factor is shown after variable ordering according to the COLAMD heuristic. The computation performed for a batch solution based on the R matrix with this ordering corresponds to the Schur complement, where all landmarks are placed at the end and therefore solved for first, corresponding to a dense matrix inversion of a constant size matrix; the remaining trajectory-only estimation can then be solved efficiently.

Updating in this case takes time $O(N)$ linear in the number of landmarks N. However, as N is constant, this corresponds to constant time $O(1)$ updates. Back-substitution requires recovery of n variables, while each row of the factor matrix contains $O(N)$ entries. Therefore, the overall complexity is $O(nN) = O(MN + N^2)$. Using the same argument as before that N is constant, we obtain linear time complexity $O(n)$ but with a potentially large constant. A batch solution based on the Schur complement takes $O(MN + N^3)$ time, where the N^3 terms arises from the dense matrix inversion over the landmarks. With the same argument as before the complexity of the batch solution reduces to linear time $O(n)$.

3.6.3 Planar Graph

Theoretical complexity bounds are available for some typical SLAM applications for which the estimation problem can be framed as a *planar graph*. While typical SLAM problems do not have a planar graph structure, for many applications such as mapping a city or exploring a building, a planar meta graph can be constructed as shown by Krauthausen *et al.* [90]. Figure 25 shows an example of such a meta graph for an indoor SLAM scenario. In particular, for a planar mapping problem with restricted sensor range we can always find such a planar meta graph. Another example is a city environment for which a meta graph is provided by a typical street map. For a planar graph, nested dissection provides a variable



(a) Typical measurement pattern.

(b) Isomorph non-planar graph.

(c) Measurement graph for a typical indoor scenario (left) and the corresponding planar meta graph (right).

Figure 25: Many SLAM scenarios are non-planar, but have a corresponding planar meta graph that allows us to establish bounds on their computational complexity (from [90]).

ordering that achieves an upper bound of $O(n \log n)$ for the fill-in of the square root factor. While this bound does not allow a statement about the maximum fill-in of an individual column of R, we can state that the average fill-in per column is bound by $O(\log n)$.

For planar graphs, iSAM updates take between $O(\log n)$ and $O(n)$ time. It is not straightforward to find the exact bound. While the new measurement row has by definition a constant number of entries, application of a Givens rotation creates new non-zero entries. In contrast to the exploration task discussed earlier, the factor matrix has $O(\log n)$ non-zero entries per column, and therefore the new measurement row has $O(\log n)$ non-zero entries after the first Givens rotation is applied. Therefore, at least $O(\log n)$ more Givens rotations are now needed to finish removing all no-zero entries in the new measurement row. However, depending on the distribution of the non-zero entries between multiple columns of the factor matrix, any further Givens rotation might again add new non-zero entries, between 0 and up to $O(\log n)$. Obviously, the maximum number of non-zero entries that could be created in the process is $O(n)$. Therefore, without further knowledge of the sparsity pattern of the R factor we cannot derive a tighter bound. Solving by back-substitution takes $O(n \log n)$ time, because n variables are recovered, each requiring an average of $O(\log n)$ substitutions based on the average number of entries in R. Furthermore, Krauthausen *et al.* [90] provides an $O(n^{1.5})$ bound on the batch solution.

3.6.4 Other Cases

The worst case scenario is a dense factor matrix, ie. a matrix in which the number of entries per column directly depends on n. For a dense factor matrix, updating takes linear time, back-substitution quadratic, and a batch solution is cubic in the number of variables. For pose constraints a completely dense information matrix and therefore also factor matrix is indeed possible if pose constraints are obtained between all pairs of robot poses, which of course would not make sense to do in the first place. We have not found a practical case in which the factor matrix actually becomes dense, partially because of the typically sparse dependencies between robot poses in both pose-only and landmark-based SLAM. However, we cannot guarantee that such a case might never arise in practice, especially also because

we use a heuristic to find a good variable ordering.

While real-world SLAM applications frequently require closing of loops, they will also constantly explore new territory, as that is the main goal of SLAM. Therefore, I do expect and have observed in my experiments that the fill-in per column is close to constant (see Figure 23), which for practical purposes includes $O(\log n)$. However, that constant depends on the "loopyness" of the environment, ie. on how spread out the reobservations of landmarks are across the full length of the robot trajectory.

Other factors also influence the computational complexity, but only by constant factors. However, that constant can of course make a huge difference in the actual execution time and therefore determine if a specific real-time application is possible or not. One such factor is the average number of landmarks visible per frame. Another one is the average number of frames for which a landmark remains visible, which is clearly connected to both vehicle speed and sensor range. Finally, the dimensionality of the variables to be estimated also influences the factor.

3.7 Related Work

SLAM algorithms only become useful for robot deployment when processing data incrementally rather than in a batch fashion. Most incremental SLAM algorithms that apply smoothing are based on iterative solvers, such as [114, 65]. The problem with iterative solvers is that there is no efficient method to recover the covariance matrix other than inverting the information matrix.

Recently, some incremental SLAM algorithms employed direct equation solvers based on Cholesky or QR factorization. Treemap by Frese [54] uses Cholesky factors to represent probability distributions in a tree-based algorithm. However, in contrast to iSAM, approximations are employed to reduce the complexity.

To the best of my knowledge, updating of matrix factorizations has not been applied in the context of SLAM before our initial work in [79]. We have since published an extended article [81] that presents parts of this chapter. There are two other works, however, that have come up with similar approaches simultaneously, and independently, of ours. One

was performed by Folkesson *et al.* [49] and uses incremental QR factorization updates for tracking features and only keeps a short history of robot poses, avoiding problems with fill-in caused by loop closing. The other can be found in the dissertation of Wang [136], where incremental Cholesky factorization is mentioned as an option for updating a factorization of the D-SLAM information matrix that only contains landmarks.

However, updating of matrix factorizations is a well-known technique in many others areas, with applications such as computer vision [134, 84] and signal processing [93]. Golub and Van Loan [60] present general methods for updating matrix factorizations based on [59, 58], including the Givens rotations I use in this work. Davis has done much research in the areas of variable ordering and factorization updates, and provides highly optimized software libraries [18, 19, 17] for various such tasks.

Maybe the first work in connection with robotics that used incremental matrix factorizations is due to Van der Merwe and Wan [97], which describes incremental updates for an unscented Kalman filter (UKF) for general state and parameter estimation, with an example application of training a neural network for "mapping the joint angles of a robot arm to the Cartesian coordinates of the hand." In the meantime, Ranganathan and Yang [119] applied our work to obtain a fast Gaussian Process inference algorithm with visual tracking applications.

Chapter IV

DATA ASSOCIATION

Real-time SLAM applications require an online solution to the data association problem. In SLAM the data association problem, which is also known as correspondence problem, consists of matching the current measurements with their corresponding previous observations. Correspondences can be obtained directly between measurements taken at different times, or by matching the current measurements to landmarks in the map that were created from earlier measurements. A solution to the correspondence problem provides frame-to-frame matching, but also allows for closing large loops in the trajectory. Loops are more difficult as the estimation uncertainty is much larger than between successive frames, and the measurements might even be taken from a different direction. The examples presented in the previous chapter made use of known correspondences. However, in order to provide a complete real-time solution to SLAM, the correspondences also have to be established in real-time.

In this chapter I show how to perform online data association with iSAM. I start with a general discussion of the data association problem and various state of the art solutions, most of which require some knowledge about the underlying estimation uncertainty. This uncertainty is represented by parts of the covariance matrix, called a marginal covariance. I present an efficient algorithm for recovering marginal covariances from the R factor, which avoids the prohibitive cost of calculating all entries of the dense covariance matrix. Beyond typical data association work, I also show how to use these marginal covariances to determine the value of a specific measurement, allowing to drop redundant or uninformative measurements in order to increase estimation efficiency. I conclude this chapter with detailed evaluations on simulated and real-world data.

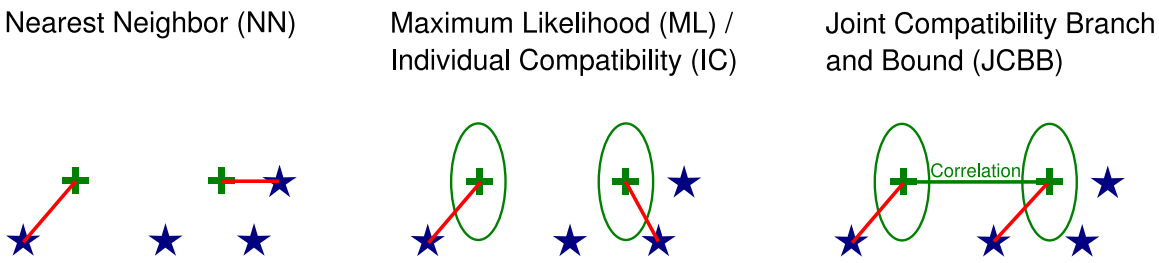


Figure 26: Comparison of some common data association techniques discussed in this chapter. Nearest neighbor assigns a measurement (blue star) to the closest landmark (green cross). Individual compatibility takes the estimation uncertainty between sensor and landmark into account, here indicated as green ellipses. The assignment is different as it now corresponds to nearest neighbor under a Mahalanobis distance. Joint compatibility branch and bound additionally takes into account the correlation between the landmarks, again yielding in a different assignment for this example.

4.1 Data Association Techniques

Next I present some of the most common data association techniques, from the trivial nearest neighbor method over the maximum likelihood solution to the joint compatibility branch and bound algorithm. I also discuss the landmark-free case, where data association is based for example on dense laser scan matching. We will see that all advanced methods have in common that they require knowledge of parts of the overall SLAM estimation uncertainty. In subsequent sections I show how to efficiently obtain these quantities using iSAM.

4.1.1 Nearest Neighbor

For completeness I include the often used *nearest neighbor* (NN) approach to data association, even though it is not sufficient for most practical applications. NN assigns each measurement to the closest landmark predicted by the current state estimate, as shown in Figure 26. Remember that the predicted measurement $\mathbf{z}_{ij}$ of landmark $\mathbf{l}_j$ taken at pose $\mathbf{x}_i$ is given by the measurement model

$$\mathbf{z}_{ij} := h_{ij}(\mathbf{x}) + \mathbf{v} \tag{22}$$

as defined in (4), where $\mathbf{v}$ is additive zero-mean Gaussian noise with covariance Γ. Note that $\mathbf{x}$ is the full state vector that includes $\mathbf{x}_i$ and $\mathbf{l}_j$.

I formulate the correspondence problem for a specific measurement k in the following way: We have an actual measurement $\tilde{\mathbf{z}}_k$ that we know was taken at time i_k and we want to determine which landmark j_k gave rise to this measurement. I define the nearest neighbor cost $D_{kj}^{2,\mathrm{NN}}$ of the hypothesis $j_k = j$ simply as the squared distance between the actual measurement $\tilde{\mathbf{z}}_k$ and its prediction $\hat{\mathbf{z}}_{i_k j} = h_{i_k j}(\hat{\mathbf{x}})$ based on the mean of the current state estimate $\hat{\mathbf{x}}$ as follows

$$D_{kj}^{2,\mathrm{NN}} := \left\| h_{i_k j}(\hat{\mathbf{x}}) - \tilde{\mathbf{z}}_k \right\|^2 . \tag{23}$$

We accept the hypothesis that landmark j gave rise to measurement $\tilde{\mathbf{z}}_k$ only if this squared distance falls below a threshold

$$D_{kj}^{2,\mathrm{NN}} < D_{\max}^2 \tag{24}$$

where the threshold $D_{\max}^2$ limits the maximum squared distance allowed between a measurement and its prediction in order to still be considered as a potential match. The threshold is typically chosen based on the nature of the data, ie. the minimum distance between landmarks as well as the expected maximum uncertainty of the measurements.

When considering multiple measurements at the same time, the NN approach can be formulated as a minimum cost assignment problem that also takes mutual exclusion into account. Mutual exclusion means that once a landmark is assigned to a measurement it is no longer available for other assignments. For K measurements and N landmarks, we form a $K \times N$ matrix that contains the cost for each possible assignment. In order to deal with unassigned measurements, which can arise from newly observed landmarks or from noise, we augment the matrix by K columns that all contain the threshold D_{max}^2

$$\mathcal{D} = \begin{bmatrix} D_{11}^2 & D_{12}^2 & D_{1N}^2 & D_{\max}^2 & \cdots \\ D_{21}^2 & \ddots & \vdots & \vdots & \\ \vdots & & & & \\ D_{K1}^2 & & D_{KN}^2 & D_{\max}^2 & \end{bmatrix} . \tag{25}$$

We use the Jonker-Volgenant-Castanon (JVC) algorithm [74] to optimally solve this assignment problem.

The biggest advantage of NN over other methods is that it does not require any knowledge of the uncertainties of the estimation problem. However, that is also its biggest weakness, as NN will eventually fail once the uncertainties become too large, as is the case when closing large loops.

4.1.2 Maximum Likelihood Data Association

The *maximum likelihood* (ML) solution to data association [5] is based on probabilistic considerations and takes into account the relative estimation uncertainties between the current robot location and the landmark in the map, as indicated in Figure 26. This technique is also known as individual compatibility (IC) [105] matching. It corresponds to the nearest neighbor method, but with the Euclidean distance replaced by the Mahalanobis distance based on the projected estimation uncertainty.

Probabilistically, the most general question we can ask about an individual measurement $\tilde{\mathbf{z}}_k$ is what the probability $P(\tilde{\mathbf{z}}_k, j_k = j|Z^-)$ is that this measurement was caused by landmark j, given all previous measurements Z^-. This expression does not yet contain any connection to the state estimate discussed in previous chapters. However, we can simply introduce the state $\mathbf{x}$, a vector that combines all landmarks $\mathbf{l}_j$ and poses $\mathbf{x}_i$, and then integrate it out again

$$\begin{aligned} P(\tilde{\mathbf{z}}_k, j_k = j|Z^-) &= \int_{\mathbf{x}} P(\tilde{\mathbf{z}}_k, j_k = j, \mathbf{x}|Z^-) \\ &= \int_{\mathbf{x}} P(\tilde{\mathbf{z}}_k, j_k = j|\mathbf{x}, Z^-)P(\mathbf{x}|Z^-) \\ &= \int_{\mathbf{x}} P(\tilde{\mathbf{z}}_k, j_k = j|\mathbf{x})P(\mathbf{x}|Z^-) \end{aligned} \tag{26}$$

where I applied the chain rule to obtain the measurement likelihood $P(\tilde{\mathbf{z}}_k, j_k = j|\mathbf{x}, Z^-) = P(\tilde{\mathbf{z}}_k, j_k = j|\mathbf{x})$, which is independent of all previous measurements Z^- given the state $\mathbf{x}$. We already know the prior $P(\mathbf{x}|Z^-)$, as that is the current state estimate, or in the case of iSAM simply a normal distribution with mean $\hat{\mathbf{x}}$ and covariance Σ. We also know the measurement likelihood $P(\tilde{\mathbf{z}}_k, j_k = j|\mathbf{x})$ as it is defined by the predictive distribution from (4). The complete probability distribution is therefore an integral over normal distributions

that can be simplified to a single normal distribution

$$P(\tilde{\mathbf{z}}_k, j_k = j|Z^-) = \int_{\mathbf{x}} \frac{1}{\sqrt{|2\pi\Gamma|}} e^{-\frac{1}{2}\left\|h_{i_k j}(\mathbf{x})-\tilde{\mathbf{z}}_k\right\|^2_\Gamma} \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}\|^2_\Sigma} \tag{27}$$
$$= \frac{1}{\sqrt{|2\pi C_{i_k j}|}} e^{-\frac{1}{2}\left\|h_{i_k j}(\hat{\mathbf{x}})-\tilde{\mathbf{z}}_k\right\|^2_{C_{i_k j}}}$$

where the covariance $C_{i_k j}$ is defined as

$$C_{i_k j} := \left.\frac{\partial h_{i_k j}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}} \Sigma \left.\frac{\partial h_{i_k j}}{\partial \mathbf{x}}\right|^T_{\hat{\mathbf{x}}} + \Gamma. \tag{28}$$

I drop the normalization factor of (27) as it does not depend on the actual measurement but is constant given the state. Further, I take the negative logarithm of the remaining expression from (27) to obtain the maximum likelihood cost function

$$D^{2,\mathrm{ML}}_{kj} := \|h_{i_k j}(\hat{\mathbf{x}}) - \tilde{\mathbf{z}}_k\|^2_{C_{i_k j}} \tag{29}$$

where again we evaluate the hypothesis that a specific measurement $\tilde{\mathbf{z}}_k$ taken in image i_k was caused by the j^{th} landmark. Note that this distance function is exactly the same as for the NN problem in (23) except for that it takes into account the uncertainties of the state estimate given by the covariance Σ. As this squared distance function follows a chi-square distribution, we base the acceptance decision

$$D^{2,\mathrm{ML}}_{kj} < \chi^2_{d,\alpha} \tag{30}$$

on the chi-square test, where α is the desired confidence level and d is the dimension of the measurement. Going back to probabilities for a moment, the threshold being exceeded means that the difference of the sample (the actual measurement $\tilde{\mathbf{z}}_k$) from the mean of the actual distribution (measurement $\hat{\mathbf{z}}_{i_k j}$ predicted by the measurement model based on the state estimate given by $\hat{\mathbf{x}}$ and Σ) is statistically significant, and we can assume that the measurement was not caused by that specific landmark as assumed by our hypothesis. For example, for a confidence level of 95% and a three dimensional measurement, the appropriate threshold is $\chi^2_{3,0.95} = 7.8147$. The relevant chi-square values are either obtained from a lookup table, or are calculated based on the incomplete gamma function and a root-finding method [116].

When considering multiple measurements simultaneously the resulting matching problem can again be reduced to a minimum cost assignment problem and solved using JVC in much the same way as was shown for NN in the previous section. More details on this minimum cost assignment problem and on how to deal with spurious measurements in a more principled probabilistic framework are provided by Dellaert [28].

For implementation purposes it is important to note that for a specific predicted measurement $\hat{\mathbf{z}}_{ij}$ of the landmark $\hat{\mathbf{l}}_j$ at the pose $\hat{\mathbf{x}}_i$, the equations above only consider a subset of the mean $\hat{\mathbf{x}}$ of the full state estimate that only contains this landmark and pose pair

$$\hat{\mathbf{x}}_{ij} = \begin{pmatrix} \hat{\mathbf{l}}_j \\ \hat{\mathbf{x}}_i \end{pmatrix}. \tag{31}$$

Similar is true for the covariance, where only the relevant components

$$\Sigma_{\mathbf{x}_{ij}} = \begin{bmatrix} \Sigma_{jj} & \Sigma_{ij}^T \\ \Sigma_{ij} & \Sigma_{ii} \end{bmatrix} \tag{32}$$

of the covariance matrix Σ of the full estimation problem are needed. iSAM provides efficient access to these quantities as I explain later in this chapter.

4.1.3 Joint Compatibility Branch and Bound (JCBB)

Instead of making independent decisions for each measurement, we now consider all measurements at the same time, which allows us to also take correlations between landmarks into account, as shown in Figure 26. This is called joint compatibility [105], and works in ambiguous configurations in which individual compatibility often fails. Such ambiguous configurations are often encountered in real-world data association problems, in particular under high motion uncertainty and when closing large loops.

This time we are interested in the probability of all measurements simultaneously given an estimate for the landmark locations as well as the robot pose. A joint hypothesis

$$\mathbf{j} = \begin{pmatrix} \mathbf{j}_1 \\ \vdots \\ \mathbf{j}_K \end{pmatrix} \tag{33}$$

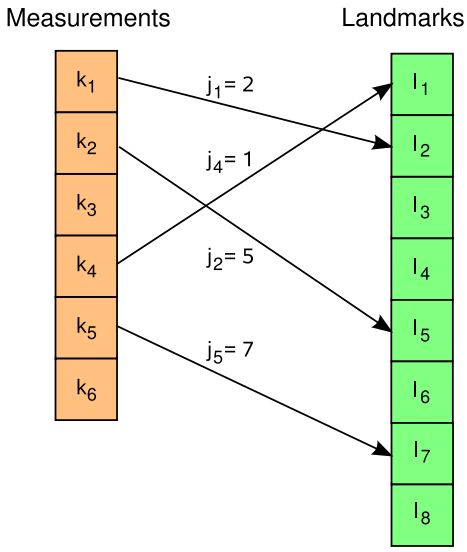


Figure 27: Example of a joint hypothesis $\mathbf{j}$ that assigns a set of measurements to landmarks. Measurements that are not assigned can be used to initialize new landmarks. Not all landmarks are necessarily visible or detected. Note that mutual exclusion prevents two measurements in the same frame from being assigned to the same landmark.

for K measurements

$$\mathbf{k} = \begin{pmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_K \end{pmatrix} \tag{34}$$

assigns each measurement $\mathbf{k}_\iota$ to a landmark $\mathbf{j}_\iota$, see Figure 27 for an example. I combine the individual measurement functions $\mathbf{z}_{i\mathbf{j}_1}, \ldots, \mathbf{z}_{i\mathbf{j}_K}$ into the joint measurement vector $\mathbf{z}_{i\mathbf{j}}$

$$\mathbf{z}_{i\mathbf{j}} := \begin{pmatrix} \mathbf{z}_{i\mathbf{j}_1} \\ \vdots \\ \mathbf{z}_{i\mathbf{j}_K} \end{pmatrix} = \begin{pmatrix} h_{i\mathbf{j}_1}(\mathbf{x}) + \mathbf{v} \\ \vdots \\ h_{i\mathbf{j}_K}(\mathbf{x}) + \mathbf{v} \end{pmatrix} = h_{i\mathbf{j}}(\mathbf{x}) + \mathbf{v}. \tag{35}$$

Analogous to the ML case we get the same expression for the probability

$$P(\tilde{\mathbf{z}}_\mathbf{k}, \mathbf{j}_\mathbf{k} = \mathbf{j} | Z^-) = \frac{1}{\sqrt{|2\pi C_{i\mathbf{j}}|}} e^{-\frac{1}{2}\left\| h_{i\mathbf{j}}(\hat{\mathbf{x}}) - \tilde{\mathbf{z}}_\mathbf{k} \right\|^2_{C_{i\mathbf{j}}}} \tag{36}$$

from (27), but the mean $\tilde{\mathbf{z}}_\mathbf{k}$ is now a joint measurement, and the distribution has covariance $C_{i\mathbf{j}}$

$$C_{i\mathbf{j}} = \left.\frac{\partial h_{i\mathbf{j}}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}} \Sigma \left.\frac{\partial h_{i\mathbf{j}}}{\partial \mathbf{x}}\right|^T_{\hat{\mathbf{x}}} + \Gamma_K \tag{37}$$

based on the Jacobian of the joint measurement function $h_{i\mathbf{j}}$ at the current state estimate $\hat{\mathbf{x}}$ with covariance Σ.

For a specific joint measurement $\tilde{\mathbf{z}}_{\mathbf{k}} := (\tilde{\mathbf{z}}_{\mathbf{k}_1}, \ldots, \tilde{\mathbf{z}}_{\mathbf{k}_K})^T$ the joint compatibility cost or squared distance function is now given by

$$D^{2,\mathrm{JC}}_{\mathbf{kj}} = \|h_{i\mathbf{j}}(\hat{\mathbf{x}}) - \tilde{\mathbf{z}}_{\mathbf{k}}\|^2_{C_{i\mathbf{j}}} . \tag{38}$$

The *joint compatibility test* for the joint hypothesis $\mathbf{j}$ is given by:

$$D^{2,\mathrm{JC}}_{\mathbf{kj}} < \chi^2_{d,\alpha} \tag{39}$$

where α is again the desired confidence level, but d is now the sum of the dimensions of all measurements that are part of the hypothesis.

For practical purposes it is important to note that the joint prediction $\mathbf{z}_{i\mathbf{j}}$ only requires all potentially observed landmarks $\hat{\mathbf{l}}_{\mathbf{j}_\iota}$ and the current pose $\hat{\mathbf{x}}_i$:

$$\hat{\mathbf{x}}_{i\mathbf{j}} = \begin{pmatrix} \hat{\mathbf{l}}_{\mathbf{j}_1} \\ \vdots \\ \hat{\mathbf{l}}_{\mathbf{j}_K} \\ \hat{\mathbf{x}}_i \end{pmatrix} \tag{40}$$

and equivalently, we only need a subset of the blocks of the full covariance matrix

$$\Sigma_{\mathbf{x}_{i\mathbf{j}}} = \begin{bmatrix} \Sigma_{\mathbf{j}_1\mathbf{j}_1} & \cdots & \Sigma^T_{\mathbf{j}_K\mathbf{j}_1} & \Sigma^T_{i\mathbf{j}_1} \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma_{\mathbf{j}_K\mathbf{j}_1} & \cdots & \Sigma_{\mathbf{j}_K\mathbf{j}_K} & \Sigma^T_{i\mathbf{j}_K} \\ \Sigma_{i\mathbf{j}_1} & \cdots & \Sigma_{i\mathbf{j}_K} & \Sigma_{ii} \end{bmatrix} \tag{41}$$

which now contains additional off-diagonal entries $\Sigma_{\mathbf{j}_\iota\mathbf{j}_{\iota'}}$ not contained in the set of individual covariances $\Sigma_{\mathbf{y}}$ for the same landmarks. These additional off-diagonal entries specify the correlation between landmarks and are essential for ambiguous situations which often arise in large loop closings.

4.1.3.1 Branch and Bound

Because the measurements are no longer independent, the search space is too large to exhaustively enumerate all possible assignments. In contrast, for NN and ML data association

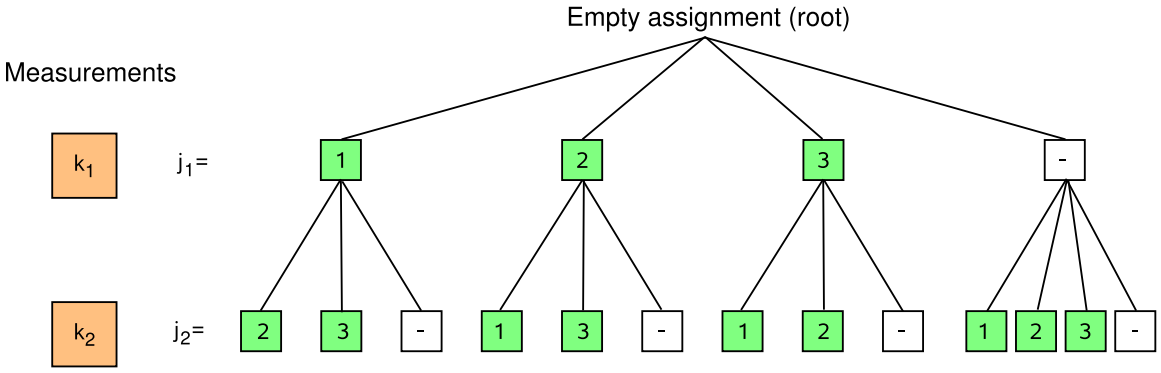


Figure 28: Tree of possible joint hypotheses for a small example with 3 landmarks (green) and 2 measurements (orange). The tree has one level per measurement, and the branching degree depends on the number of landmarks. Note that a measurement can remain unassigned, indicated by a dash ("-"), and that mutual exclusion is enforced.

we simply enumerate each possible landmark-to-feature assignment. For K features and N landmarks there are

$$\Pi^{\text{individual}} = KN \tag{42}$$

different costs to evaluate. Only then do we deal with unassigned measurements and mutual exclusion by means of the minimum cost assignment problem. For joint compatibility, however, the measurements are not independent and we therefore have to evaluate the cost of each joint hypothesis that either assigns each feature to a landmark or leaves it unassigned. The overall number of these joint hypotheses is far larger than the number of individual hypotheses. In particular, we can assign one of N landmarks to the first feature or leave it unassigned, and at the same time one of the $N-1$ remaining ones to the second (considering mutual exclusion) or leave it unassigned and so on, yielding

$$\Pi^{\text{joint}} = \frac{(N+1)!}{(N+1-K)!} \tag{43}$$

possible joint hypotheses to evaluate, which is $O(N^K)$.

The combinatorial complexity of the state space is addressed by the joint compatibility branch and bound (JCBB) algorithm by Neira and Tardos [105]. *Branch and bound* represents the space of all possible hypotheses by a tree, as shown in Figure 28, where each node represents a specific hypothesis. The goal of the algorithm is to find the hypothesis with the highest number of jointly compatible matchings at the lowest squared distance $D^{2,\text{JC}}_{\mathbf{kj}}$

```
# general branch and bound algorithm
# the input arguments are functions tailored to the specific application
function branch_and_bound(successors, bound, prunable,
                          cost, feasible)
  best = None
  heap = insert(empty_node, bound(empty_node), empty_heap)
  while not empty(heap)
    # process next node
    node = get_min(heap)
    # skip subtree if bound is worse than best
    if not prunable(node, best) then
      # update best if feasible and better than current best
      if feasible(node) and cost(node) < cost(best) then
        best = node
      endif
      # insert successor nodes
      for all successors(node) do
        insert(successor, bound(successor), heap)
      done
    endif
  endwhile
  return best
```

Figure 29: The general branch and bound algorithm. The functions needed for the joint compatibility branch and bound (JCBB) algorithm are described in the text.

according to (38).

The algorithm starts with an empty hypothesis, the root of the tree, and processes nodes from a queue starting from the most promising one as determined by a lower bound of the cost. For this purpose the algorithm adds to the queue new hypotheses for successors of nodes that it encounters. Efficiency is achieved by discarding subtrees whose lower bound is higher than the current best hypothesis. The branch and bound algorithm is shown in Figure 29. For the JCBB algorithm the following functions have to be defined:

`successors` returns a list of hypotheses, for which the next unprocessed measurement is assigned to any of the valid landmarks or alternatively left unassigned. However, only assignments that fulfill the individual compatibility test from (30) are acceptable. Further, the mutual exclusion constraint is enforced by not considering landmarks that have already been assigned in this hypothesis.

`cost` returns the cost of a hypothesis, which is primarily defined by the number of assignments, where a higher number of assignments yields a lower cost. The squared distance $D^{2,\mathrm{JC}}_{\mathbf{kj}}$ from (38) is used to further differentiate between hypotheses with equal numbers of assignments.

`feasible` returns true if the hypothesis represents an acceptable assignment. An assignment is acceptable if the cost of the node is lower than $\chi^2_{d,\alpha}$ following (39), ie. this is the actual joint compatibility test.

`bound` returns a bound on the cost of any node in the subtree rooted at this node. The bound is defined by the cost of the node, but in addition to the current number of assignments we also consider the number of measurements that are still unprocessed – none of the children can possibly achieve a lower cost.

`prunable` returns true if the subtree rooted at this node can be discarded. A subtree can be discarded if its bound is higher than the cost of the current best hypothesis.

Note that JCBB might still become very expensive when a large number of landmarks and measurements are considered. In practice it is advisable to only match against potentially

visible landmarks, see also Section 4.1.4. To further reduce costs, the joint compatibility has to be evaluated incrementally as described next.

4.1.3.2 Incrementally Calculating Joint Compatibility

For efficiency, the joint compatibility criterion has to be evaluated incrementally to avoid inversion of increasingly large matrices during branch and bound. Please see the original paper by Neira and Tardos [105] for the mathematical derivation. The goal is to incrementally calculate the squared distance

$$D^{2,\mathrm{JC}} = h^T C^{-1} h \tag{44}$$

where

$$C = H\Sigma H^T + \Gamma \tag{45}$$

and

$$h := h(\hat{\mathbf{x}}) - \tilde{\mathbf{z}} \tag{46}$$

with

$$H := \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}} \tag{47}$$

by using previously calculated components to avoid the inversion of an increasingly large C matrix. Assume we have $D^{2,\mathrm{JC}}_-$, C^{-1}_-, h_-, H_- from previous calculations, where the subscript "$-$" refers to the current hypothesis represented by $\mathbf{j} = (\mathbf{j}_1, \ldots, \mathbf{j}_K)^T$ but with the last entry $\mathbf{j}_K$ unassigned, and I use the subscript "$+$" for the hypothesis that only contains this last assignment $\mathbf{j}_K$ and all other measurements are unassigned. Let us define

$$w := H_+ \Sigma H \tag{48}$$

$$N := \left(C_+ - wC^{-1}_- w^T\right)^{-1} \tag{49}$$

$$L := -NwC^{-1}_- \tag{50}$$

$$K := C^{-1}_- + L^T N^{-1} L \tag{51}$$

where I assume that Γ is (block-)diagonal to simplify the expression of w. The squared distance for the full assignment can now be calculated incrementally:

$$D^{2,\mathrm{JC}} = D^{2,\mathrm{JC}}_- + h^T_- L^T N^{-1} L h_- + 2h^T_+ L h_- + h^T_+ N h_+ \tag{52}$$

and we additionally need to remember

$$C^{-1} = \begin{bmatrix} K & L^T \\ L & N \end{bmatrix} \tag{53}$$

$$H = \begin{bmatrix} H_- \\ H_+ \end{bmatrix} \tag{54}$$

$$h = \begin{pmatrix} h_- \\ h_+ \end{pmatrix} \tag{55}$$

for future calculations.

4.1.3.3 Simplified JCBB

While I am not using the *simplified* joint compatibility branch and bound algorithm by Clemente *et al.* [14] in this work, I mention it here as an interesting, but much cheaper alternative to the full algorithm. The simplified JCBB algorithm starts from a known assignment vector and makes a binary decision for each single assignment. The full JCBB algorithm can be quite expensive. Even though large parts of the search tree are pruned by bounding the cost, the branching factor of a single node is large in practical applications. Other algorithms, such as ML, can provide assignment vectors that are often correct at very low cost because they ignore joint compatibility. The simplified JCBB algorithm is then used to identify and potentially correct wrong assignments.

Simplified JCBB starts with an hypothesis $\mathbf{j}$ of assignments between measurements and landmarks. The hypothesis is then tested for joint compatibility. Branch and bound is only performed if the test fails in order to find the largest set of jointly compatible assignments in $\mathbf{j}$, where each measurement either remains assigned to its original landmark, or is changed to unassigned status. Note that branch and bound is still important here because a sequential decision of which assignment is acceptable would depend on the order in which the assignments are processed. Of course, under high uncertainty, most or all assignments made by ML might be incorrect, and the simplified JCBB algorithm will not be able to find a correct assignment. However, at least such a failure can then be identified.

4.1.4 Search Region and Pose Uncertainty

How far do we need to search for potential loop closures? We face this question for example for landmark-based data association, as we want to quickly identify landmarks that are potential candidates for a match in a given correspondence decision. The same problem also appears in place recognition, such as our work in [102], where restricting the search region allows keeping the computational requirements of place recognition low even in very large-scale applications. Furthermore, the problem appears in pose-only estimation, such as laser-based scan matching, where we need to identify earlier parts of the trajectory (and therefore parts of the map) that are candidates for a loop closing. While I have not done so in this work (the scan matching based datasets in Chapter 3 were preprocessed and have known data association), I at least want to discuss how iSAM is also helpful in this case.

The question about restricting the search region can again be answered by recovering parts of the covariance matrix. In the simplest case we only recover the uncertainty of the current pose, which is the bottom right-most block of the full covariance matrix and therefore trivial to recover from the square root factor R. However, imagine a vehicle driving in a straight line for a long distance and then performing a small loop. The absolute pose uncertainty is very large as we are far from the starting point. This would result in a large search region for loop closing. However, the actual uncertainty within the small loop is much smaller and is obtained by taking into account the marginal covariances between the current pose and some previous pose along the loop. The off-diagonal blocks describe the correlation between poses, and therefore the search region will now be much smaller.

Analogous to the ML data association case, the question to ask is how likely it is that two poses $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are identical given all measurements Z that we have seen so far. With the same argument as before we obtain

$$\begin{aligned} P(\mathbf{x}_i = \mathbf{x}_{i'}|Z) &= \int_x P(\mathbf{x}_i = \mathbf{x}_{i'}|\mathbf{x}Z)P(\mathbf{x}|Z) \\ &= \int_x \frac{1}{\sqrt{|2\pi\Lambda|}} e^{-\frac{1}{2}\|f(\mathbf{x}_{i'},0)-\mathbf{x}_i\|^2_\Lambda} \frac{1}{\sqrt{|2\pi\Sigma|}} e^{-\frac{1}{2}\|\mathbf{x}-\hat{\mathbf{x}}\|^2_\Sigma} \\ &= \frac{1}{\sqrt{|2\pi C_{ii'}|}} e^{-\frac{1}{2}\|f(\hat{\mathbf{x}}_{i'},0)-\hat{\mathbf{x}}_i\|^2_{C_{ii'}}} \end{aligned} \tag{56}$$

where f is the process model from (2) and the covariance $C_{ii'}$ is defined as

$$C_{ii'} := \left.\frac{\partial f}{\partial x}\right|_{\hat{\mathbf{x}}} \Sigma \left.\frac{\partial f}{\partial x}\right|_{\hat{\mathbf{x}}}^{T} + \Lambda. \tag{57}$$

4.2 Selecting Informative Measurements

Which measurements provide the most information about the state estimate? The answer certainly depends on the application requirements in terms of processing speed, and there is a tradeoff between omitting information and the quality of the estimation result. However, often there are also redundant or uninformative measurements that do not add any valuable information and can therefore safely be discarded. An answer to this question allows us to only use informative measurements in the estimation process, reducing computational complexity. It can also be used to guide the search for measurements, as exploited by Davison [22] for active search, but that is not the goal of my work. In this section I discuss the theory of how to determine the information that a measurement contributes following the work by Davison [22], but diverging in important points.

Let us start by identifying from a set of measurements the single measurement that, if applied, results in the state estimate with lowest uncertainty, or in other words, results in the highest gain in information. The current state estimate is given by the probability distribution $P(\mathbf{x}|Z^-)$ over the state $\mathbf{x}$ given all previous measurements Z^-. The posterior $P(\mathbf{x}|\mathbf{z}_j, Z^-)$ provides the distribution over the state $\mathbf{x}$ after applying a measurement $\mathbf{z}_j$ on landmark j. The measurement $\mathbf{z}_j$ follows the distribution $\mathbf{z}_j = h_{ij}(\mathbf{x}) + \nu$ from (4). Note our new notation $\mathbf{z}_j$: We do not use a specific measurement, but rather the expected measurement $\hat{\mathbf{z}}_j = h_{ij}(\hat{\mathbf{x}}) + \nu$ for a *specific landmark* j given the state estimate $\hat{\mathbf{x}}$, as we want to identify the measurement that is expected to yield the largest reduction in uncertainty *before* the measurement is actually made. The expected reduction in uncertainty or gain in information about the state $\mathbf{x}$ when making measurement $\mathbf{z}_j$ is given by the mutual information [16]

$$\begin{aligned} I(\mathbf{x}; \mathbf{z}_j) &:= H(\mathbf{x}) - H(\mathbf{x}|\mathbf{z}_j) \\ &= \int_{\mathbf{x},\mathbf{z}_j} P(\mathbf{x}, \mathbf{z}_j) \log \frac{P(\mathbf{x}, \mathbf{z}_j)}{P(\mathbf{x})P(\mathbf{z}_j)}, \end{aligned} \tag{58}$$

where $H(\mathbf{x})$ is the entropy of the current state and $H(\mathbf{x}|\mathbf{z}_j)$ is the conditional entropy of the state after applying the measurement.

As iSAM represents Gaussian distributions, we obtain an expression for mutual information for this special case. We start with a random variable $\mathbf{a}$ with Gaussian distribution

$$P(\mathbf{a}) = \frac{1}{\sqrt{|2\pi\Sigma_{\mathbf{a}}|}} e^{-\frac{1}{2}\|\mathbf{a}-\hat{\mathbf{a}}\|^2_{\Sigma_{\mathbf{a}}}} \tag{59}$$

that is partitioned into two random variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ so that the joint mean $\hat{\mathbf{a}}$ and covariance $\Sigma_{\mathbf{a}}$ are given by

$$\hat{\mathbf{a}} = \begin{pmatrix} \hat{\boldsymbol{\alpha}} \\ \hat{\boldsymbol{\beta}} \end{pmatrix}, \quad \Sigma_{\mathbf{a}} = \begin{bmatrix} \Sigma_{\boldsymbol{\alpha}\boldsymbol{\alpha}} & \Sigma_{\boldsymbol{\alpha}\boldsymbol{\beta}} \\ \Sigma_{\boldsymbol{\beta}\boldsymbol{\alpha}} & \Sigma_{\boldsymbol{\beta}\boldsymbol{\beta}} \end{bmatrix} \tag{60}$$

It is shown by Davison [22] that the mutual information between the two Gaussian distributions $P(\boldsymbol{\alpha})$ and $P(\boldsymbol{\beta})$ is given by

$$\begin{aligned} I(\boldsymbol{\alpha};\boldsymbol{\beta}) &= E\left[\log_2 \frac{P(\boldsymbol{\alpha}|\boldsymbol{\beta})}{P(\boldsymbol{\alpha})}\right] \\ &= \frac{1}{2}\log_2 \frac{|\Sigma_{\boldsymbol{\alpha}\boldsymbol{\alpha}}|}{\left|\Sigma_{\boldsymbol{\alpha}\boldsymbol{\alpha}} - \Sigma_{\boldsymbol{\alpha}\boldsymbol{\beta}}\Sigma^{-1}_{\boldsymbol{\beta}\boldsymbol{\beta}}\Sigma_{\boldsymbol{\beta}\boldsymbol{\alpha}}\right|}. \end{aligned} \tag{61}$$

Note that the result is given in *bits* as we use the logarithm base 2.

To obtain the mutual information between the state space $\mathbf{x}$ and any of the N measurement functions $\mathbf{z}_j$ we combine these random variables into a new one

$$\mathbf{w} = \begin{pmatrix} \mathbf{x} \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{pmatrix} \tag{62}$$

that follows a Gaussian distribution with mean $\hat{\mathbf{w}} = (\hat{\mathbf{x}}, \hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_N)^T$ and covariance

$$\Sigma_{\mathbf{w}} = \begin{bmatrix} \Sigma & \Sigma\frac{\partial h_1}{\partial \mathbf{x}}^T & \cdots & \Sigma\frac{\partial h_N}{\partial \mathbf{x}}^T \\ \frac{\partial h_1}{\partial \mathbf{x}}\Sigma & \frac{\partial h_1}{\partial \mathbf{x}}\Sigma\frac{\partial h_1}{\partial \mathbf{x}}^T + \Gamma_1 & \cdots & \frac{\partial h_1}{\partial \mathbf{x}}\Sigma\frac{\partial h_N}{\partial \mathbf{x}}^T \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_N}{\partial \mathbf{x}}\Sigma & \frac{\partial h_N}{\partial \mathbf{x}}\Sigma\frac{\partial h_1}{\partial \mathbf{x}}^T & \cdots & \frac{\partial h_N}{\partial \mathbf{x}}\Sigma\frac{\partial h_N}{\partial \mathbf{x}}^T + \Gamma_N \end{bmatrix} \tag{63}$$

where some of the terms were defined in (28) based on (27), and the remaining ones follow by analogy.

It is now easy to select the *best* measurement by calculating all $I(\mathbf{x}; \mathbf{z}_j)$, but how do we treat the remaining measurements? We identify three different possibilities:

1. We calculate the mutual information between the state vector and all possible combinations of measurements. This is the correct solution as the measurements are not independent. However, unless the number of measurements is very small this solution is infeasible because the number of possible combinations 2^N is exponential in the number of landmarks N that can be measured.

2. We select the best measurement, then use the measurement to update the state space and start the feature selection again with the remaining measurements as done in [22]. Instead of taking the expected reduction in uncertainty into account, this solution uses the actual measurement to update the state space. However, the decision is sequential and therefore not guaranteed to be optimal. This solution is practical if updating the state space and recovering the necessary covariances are cheap operations.

3. We select the best measurement without updating the state space, and then ask the same question as before: Which of the remaining measurements is expected to yield the lowest uncertainty? This avoids the combinatorial complexity as well as updating of the state space with a measurement. While this solution is also not optimal, it is much cheaper than the other solutions. I make use of this approach as described next.

One idea for finding the best measurement from the remaining ones is to look at the mutual information between measurements in order to decide which ones are redundant, as suggested in [22]. However, from these quantities we cannot directly calculate the correct information gains, as they do not consider mutual information between combinations of measurements.

Instead, I calculate the mutual information $I(\mathbf{x}; \mathbf{z}, \mathbf{z}_1)$ of the state space $\mathbf{x}$ and measurements $\mathbf{z}_1$ and $\mathbf{z}$ for each remaining measurement $\mathbf{z}$. After selecting a measurement $\mathbf{z}_2$ I continue with the mutual information $I(\mathbf{x}; \mathbf{z}, \mathbf{z}_1, \mathbf{z}_2)$ for the remaining measurements and so on. My approach requires the same quantities as in [22], although the calculations

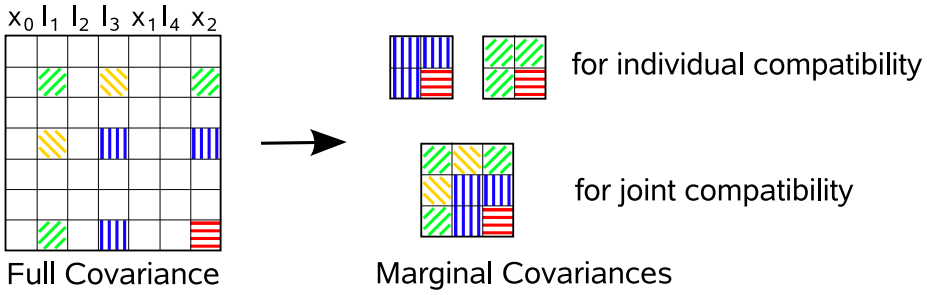


Figure 30: Only a small number of entries of the dense covariance matrix are of interest for data association. In this example, both the individual and the combined marginals between the landmarks $\mathbf{l}_1$ and $\mathbf{l}_3$ and the latest pose $\mathbf{x}_2$ are retrieved. As I show here, these entries can be obtained without calculating the full dense covariance matrix.

get slightly more expensive as increasingly large blocks from the covariance matrix $\Sigma_{\mathbf{w}}$ are needed. However, the increase in cost is not significant because the size of the state space $\mathbf{x}$ is of the same order as the size of all measurements together. Note that my approach correctly takes care of redundant features.

We are not only interested in the order of the measurements in terms of their value for the estimation process, but also want to omit measurements that are not informative enough. For that purpose we ignore features that fall below a certain threshold, for example 2 bits.

For practical purposes we again only need the subset $\hat{\mathbf{x}}_{i\mathbf{j}}$ of the mean of the full state estimate $\hat{\mathbf{x}}$ that only contains the components for visible landmarks $\hat{\mathbf{l}}_j$ and the current pose $\hat{\mathbf{x}}_i$. Similarly, only a subset $\Sigma_{\mathbf{x}_{i\mathbf{j}}}$ of the blocks of the full covariance matrix are needed

$$\hat{\mathbf{x}}_{i\mathbf{j}} = \begin{pmatrix} \hat{\mathbf{l}}_{\mathbf{j}_1} \\ \vdots \\ \hat{\mathbf{l}}_{\mathbf{j}_K} \\ \hat{\mathbf{x}}_i \end{pmatrix}, \qquad \Sigma_{\mathbf{x}_{i\mathbf{j}}} = \begin{bmatrix} \Sigma_{\mathbf{j}_1\mathbf{j}_1} & \cdots & \Sigma^T_{\mathbf{j}_K\mathbf{j}_1} & \Sigma^T_{i\mathbf{j}_1} \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma_{\mathbf{j}_K\mathbf{j}_1} & \cdots & \Sigma_{\mathbf{j}_K\mathbf{j}_K} & \Sigma^T_{i\mathbf{j}_K} \\ \Sigma_{i\mathbf{j}_1} & \cdots & \Sigma_{i\mathbf{j}_K} & \Sigma_{ii} \end{bmatrix}. \tag{64}$$

Note that these are the same quantities that were already used for joint compatibility in (40) and (41). In the next section I discuss how to efficiently obtain these entries from iSAM.

4.3 Marginal Covariances from iSAM

Knowledge of the relative uncertainties between the current pose $\mathbf{x}_i$ and any visible landmark $\mathbf{l}_j$ is needed for all but the NN solution to the data association problem. The marginal covariances

$$\begin{bmatrix} \Sigma_{jj} & \Sigma_{ij}^T \\ \Sigma_{ij} & \Sigma_{ii} \end{bmatrix}, \qquad \begin{bmatrix} \Sigma_{j_1 j_1} & \cdots & \Sigma_{j_K j_1}^T & \Sigma_{i j_1}^T \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma_{j_K j_1} & \cdots & \Sigma_{j_K j_K} & \Sigma_{i j_K}^T \\ \Sigma_{i j_1} & \cdots & \Sigma_{i j_K} & \Sigma_{ii} \end{bmatrix} \tag{65}$$

needed above for individual and joint compatibility as well as for information theoretic decisions about the value of a measurement contain various blocks from the full covariance matrix, as is shown in Figure 30. The diagonal blocks contain the uncertainties of individual variables. However, the off-diagonal blocks are essential, because the uncertainties are relative to an arbitrary reference frame, which is often fixed at the origin of the trajectory.

An example makes it easy to see why the off-diagonal entries of the covariance matrix are important for data association. Imagine a robot driving down a long hallway. The uncertainty of the current robot location, and therefore also of nearby landmarks, will continuously grow, as they are measured with respect to the starting point of the trajectory, and no loop closing takes place. Imagine further that the robot now performs a small loop at the end of this long trajectory. The absolute uncertainties are very high, making data association decisions very ambiguous. However, the actual uncertainty of this small loop is fairly small and data association should be simple. We obtain these actual uncertainties by taking the relative uncertainties between robot poses (and landmarks) into account, which are represented by the off-diagonal blocks in the covariance matrix.

Calculating the full covariance matrix to recover these entries is not an option because the covariance matrix is always densely populated with n^2 entries, where n is the number of variables. However, I show in the next section that it is not necessary to calculate all entries in order to retrieve the exact values of the relevant blocks. Further, for the ML data association case, we can also obtain conservative estimates as I show in Section 4.3.2.

4.3.1 Exact Marginal Covariances

Recovering the exact values for all required entries without calculating the complete covariance matrix is not straightforward, but can be done efficiently by again exploiting the sparsity structure of the factor matrix R. In general, the covariance matrix is obtained as the inverse of the information matrix

$$\Sigma := (A^T A)^{-1} = (R^T R)^{-1} \tag{66}$$

based on the factor R by noting that

$$R^T R \Sigma \;\; = \;\; I \tag{67}$$

and performing a forward, followed by a back-substitution

$$R^T Y = I, \quad R\Sigma = Y. \tag{68}$$

Because the information matrix is not band-diagonal in general, this would seem to require calculating all n^2 entries of the fully dense covariance matrix, which is infeasible for any non-trivial problem. This is where the sparsity of the factor R is of advantage again. Both, Golub and Plemmons [61] and Triggs *et al.* [134] present an efficient method for recovering only the entries σ_{ij} of the covariance matrix Σ that coincide with non-zero entries in the factor $R = (r_{ij})$:

$$\sigma_{ll} = \frac{1}{r_{ll}}(\frac{1}{r_{ll}} - \sum_{j=l+1, r_{lj}\neq 0}^{n} r_{lj}\sigma_{jl}) \tag{69}$$

$$\sigma_{il} \;\; = \;\; \frac{1}{r_{ii}}(- \sum_{j=i+1, r_{ij}\neq 0}^{l} r_{ij}\sigma_{jl} - \sum_{j=l+1, r_{ij}\neq 0}^{n} r_{ij}\sigma_{lj}) \tag{70}$$

for $l = n, \ldots, 1$ and $i = l-1, \ldots, 1$, where the other half of the matrix is given by symmetry. Note that the summations only apply to non-zero entries of single columns or rows of the sparse matrix R. This means that in order to obtain the top-left-most entry of the covariance matrix, we at most have to calculate all other entries that correspond to non-zeros in R. The algorithm has $O(n)$ time complexity for band-diagonal matrices and matrices with only a small number of entries far from the diagonal, but can be more expensive for general sparse

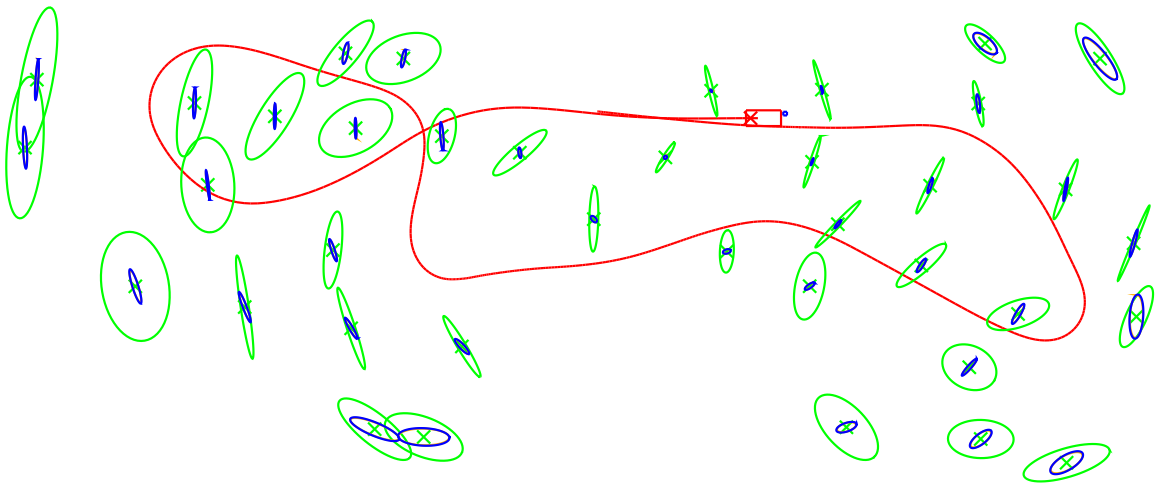


Figure 31: Comparison of marginal covariance estimates *projected into the current robot frame* (robot indicated by red rectangle) for a short trajectory (red curve) and some landmarks (green crosses). Conservative covariances (green, large ellipses) as well as the exact covariances (blue, smaller ellipses) obtained by my fast algorithm are shown. Note that the exact covariances based on full inversion are also shown (orange, mostly hidden by blue).

R. In particular, if the otherwise sparse matrix contains a dense block of side length s, the complexity is $O(nk + s^3)$ and the constant factor s^3 can become dominant for practical purposes.

Based on a dynamic programming approach, my algorithm as shown in Figure 32 provides access to all entries of interest for data association. While most of the upper triangular blocks of R are fully populated, any required entries that correspond to zeros in R are automatically obtained. An example of how the recovery proceeds is shown in Figure 33. Figure 31 shows the marginal covariances obtained by this algorithm for the first part of the Victoria Park sequence. Note that they coincide with the exact covariances obtained by full matrix inversion.

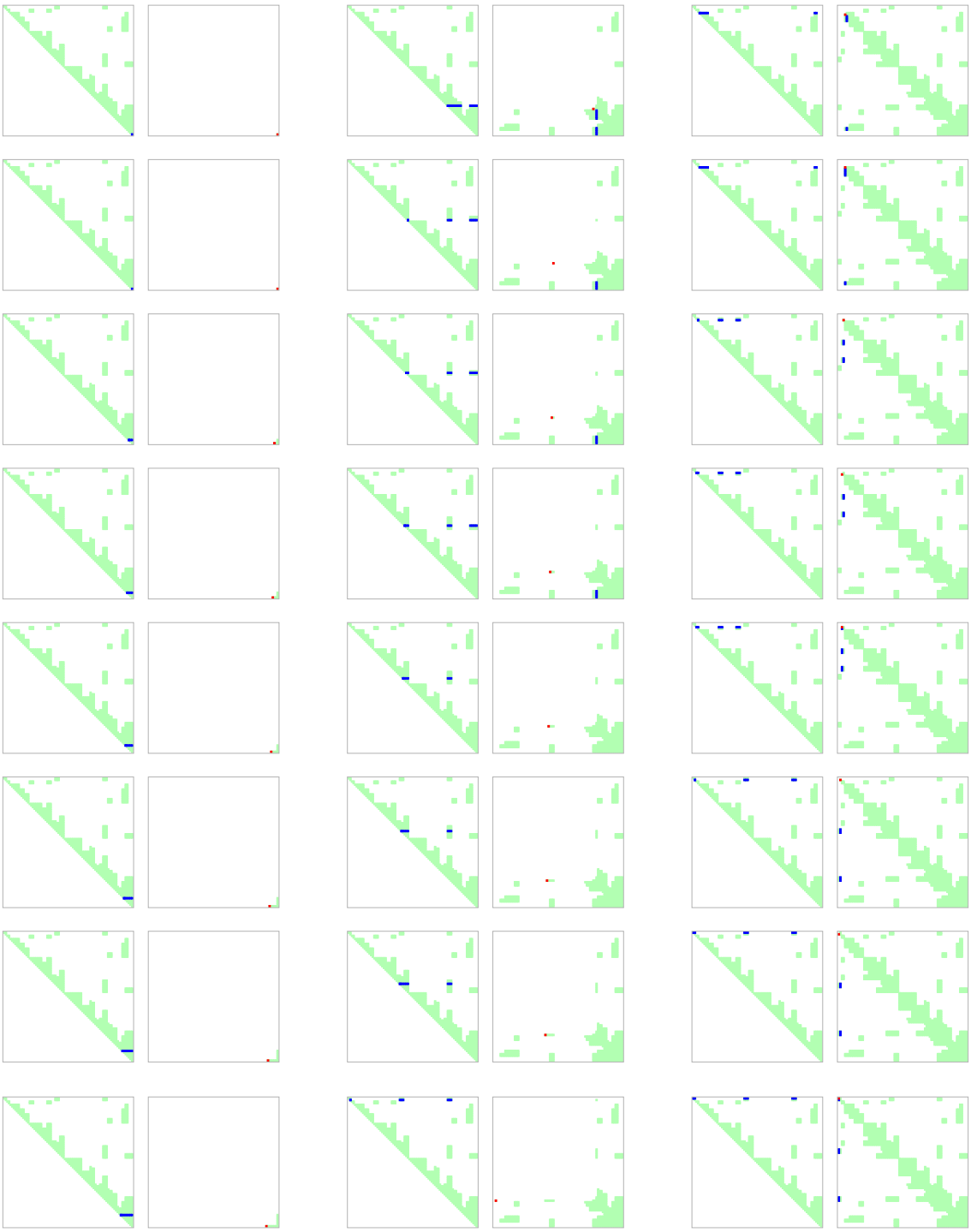
4.3.2 Conservative Estimates

For ML data association in particular, a cheap substitute for my exact covariance recovery is obtained by using conservative estimates of the block diagonal entries. The exact values of the off-diagonal blocks that are related to the current pose can be recovered using back-substitution, because I always add the most recent robot pose at the end.

The recovery of the last columns of the covariance matrix from the square root factor can

```
# recover marginal covariance of variables given by indices
function recover(R, indices) =
  n = rows(R)
  for i=1:n do  # precalculate, needed multiple times
    diag[i] = 1 / R[i,i]
  done
  # recover one entry, hashed for fast random access
  function hash entry(i, l) =
    # sum over sparse entries of one row
    function sum_j(i) =
      sum = 0
      for each entry rij of sparse row i of R do
        if j<>i then
          if j>l then
            lj = entry(l, j)
          else
            lj = entry(j, l)
          endif
          sum += rij * lj
        endif
      done
      return sum
    if i = l then    # treat diagonal entries differently
      return (diag[l] * (diag[l] - sum_j(l)))
    else
      return (- sum_j(i) * diag[i])
    endif
  # recover square marginal covariance matrix
  n_indices = length(indices)
  for r=1:n_indices do
    for c=1:n_indices do
      P[r,c] = entry(indices[r],indices[c])
    done
  done
  return P
```

Figure 32: Dynamic programming algorithm for marginal covariance recovery inspired by Golub's partial sparse matrix inversion algorithm. The argument `indices` provides the indices of variables for which the marginal covariance should be recovered. I have chosen to use a hash table for fast random access to the sparse entries calculated by `recover`. For practical implementations, care also has to be taken to avoid stack overflows caused by the recursive calls. For example if we first ask for the left top-most entry of the covariance matrix, all other required entries (the ones that correspond to non-zeros in R) will be calculated recursively. It is therefore advantageous to process entries starting from the right-most column, while taking the variable ordering into account.



(a) First 8 steps. (b) Steps 245 to 252. (c) Last 8 steps.

Figure 33: The process of recovering marginal covariances: The three columns show successive steps of recovering the entries of the covariance matrix that correspond to non-zero entries in the R factor. The overall process takes 716 steps. For each step the R factor is shown in the left half and the partially computed covariance matrix in the right half. The matrix entry to be calculated is shown in red, while blue are accessed entries and green are the remaining non-zero entries.

be done efficiently by back-substitution. The exact pose uncertainty Σ_{ii} and the covariances Σ_{ij} can be recovered in linear time, based on the sparse R factor. As I choose the current pose to be the last variable in the factor R, the last block column X of the full covariance matrix $(R^T R)^{-1}$ contains Σ_{ii} as well as all Σ_{ij} as observed in [42], but instead of having to keep an incremental estimate of these quantities, we retrieve the exact values efficiently from the factor R by back-substitution. With $d_{\mathbf{x}}$ the dimension of the last pose, I define $B \in \mathbb{R}^{n \times d_{\mathbf{x}}}$ as the last $d_{\mathbf{x}}$ unit vectors

$$B = \begin{bmatrix} 0_{(n-d_{\mathbf{x}}) \times d_{\mathbf{x}}} \\ I_{d_{\mathbf{x}} \times d_{\mathbf{x}}} \end{bmatrix} \tag{71}$$

and solve

$$R^T R X = B \tag{72}$$

by a forward and a back-substitution

$$R^T Y = B, \quad RX = Y. \tag{73}$$

The key to efficiency is that we never have to recover a full dense matrix, but due to R being upper triangular immediately obtain

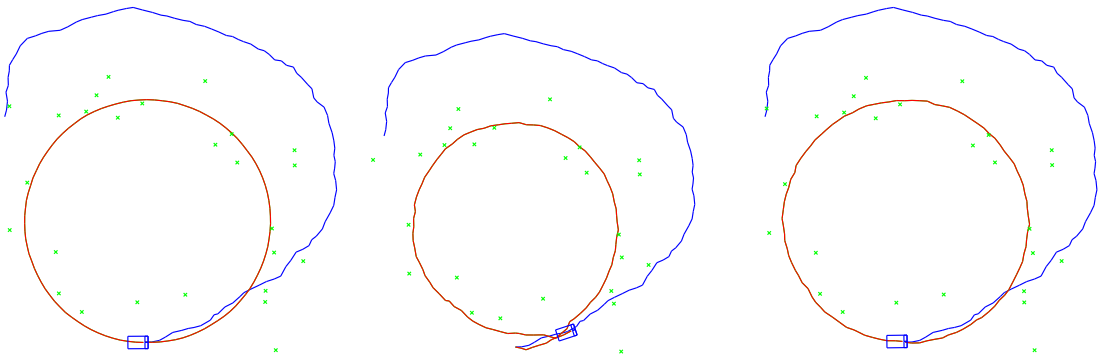
$$Y = [0, ..., 0, R_{ii}^{-1}]^T. \tag{74}$$

Recovering these columns is efficient, because only a constant number of $d_{\mathbf{x}}$ back-substitutions are needed, each of which is performed in linear time because of the sparsity of R.

Conservative estimates for the structure uncertainties Σ_{jj} are obtained from the initial uncertainties as proposed by Eustice [42]. As the uncertainty can never grow when new measurements are added to the system, the initial uncertainties $\tilde{\Sigma}_{jj}$ provide conservative estimates. These are obtained by

$$\tilde{\Sigma}_{jj} = \bar{J} \begin{bmatrix} \Sigma_{ii} & \\ & \Gamma \end{bmatrix} \bar{J}^T \tag{75}$$

where $\bar{J}$ is the Jacobian of the linearized back-projection function (an inverse of the measurement function is not always available, for example for the bearing-only case), and Σ_{ii}



(a) Ground truth. (b) ML data association fails. (c) JCBB succeeds despite high noise.

Figure 34: A simulated loop with high noise that requires the JCBB algorithm for successful data association.

and Γ are the current pose uncertainty and the measurement noise, respectively. Figure 31 provides a comparison of the conservative and exact covariances. A more tight conservative estimate on a landmark can be obtained after multiple measurements are available, or later in the process by means of the exact algorithm from the previous section.

4.4 Experiments and Results

I present timing results for recovering exact marginal covariances as well as conservative estimates. I analyze data association for the Victoria Park dataset, shown in Figure 17, which was discussed in more detail in Section 3.5.1 for the case of known correspondences. And I analyze the effect of measurement selection based on expected information gain.

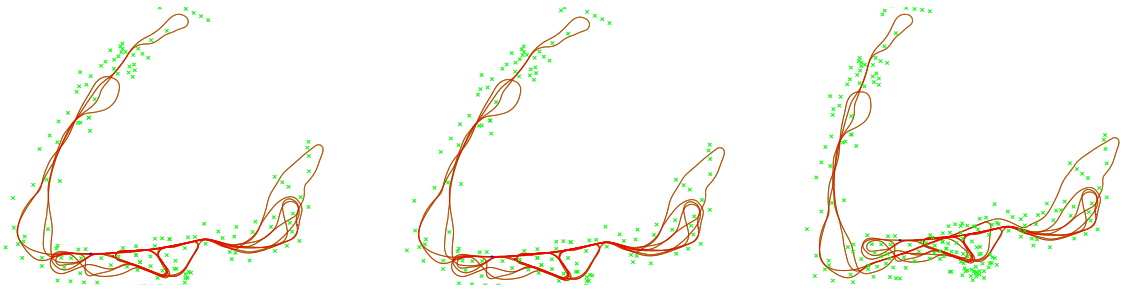
4.4.1 Exact Marginals Covariances

To show the merits of JCBB, I use a simulated environment with 100 poses, 27 landmarks and 508 measurements in Figure 34. The trajectory length is about $50m$ and all standard deviations are $0.1m$ and $0.1rad$. Maximum likelihood data association fails to successfully close the loop, as a wrong data association decision is made. JCBB on the other hand successfully establishes the correct correspondences.

Table 2 shows execution times for JCBB data association applied to the Victoria Park sequence, with different thresholds on the information content that a measurement needs to provide in order to be added to the estimation problem. For reference, I also show

Table 2: Execution times for the Victoria Park sequence under different data association techniques. The number of measurements declines with increasing threshold on the minimum information a measurement has to provide, until data association eventually fails as shown in Figure 35.

	Execution time	Number of measurements
NN	$207s$	3640
JCBB	$590s$	3640
JCBB, 2 bit threshold	$588s$	3628
JCBB, 3 bit threshold	$493s$	2622
JCBB, 4 bit threshold	fails	-



(a) 2 bit threshold (b) 3 bit threshold (c) 4 bit threshold

Figure 35: Maps resulting from omitting measurements with expected information below a threshold according to Table 2. For 2 and 3 bit thresholds the maps are visually identical. For a 4 bit threshold data association fails in the rightmost part of the trajectory, yielding an inconsistent map.

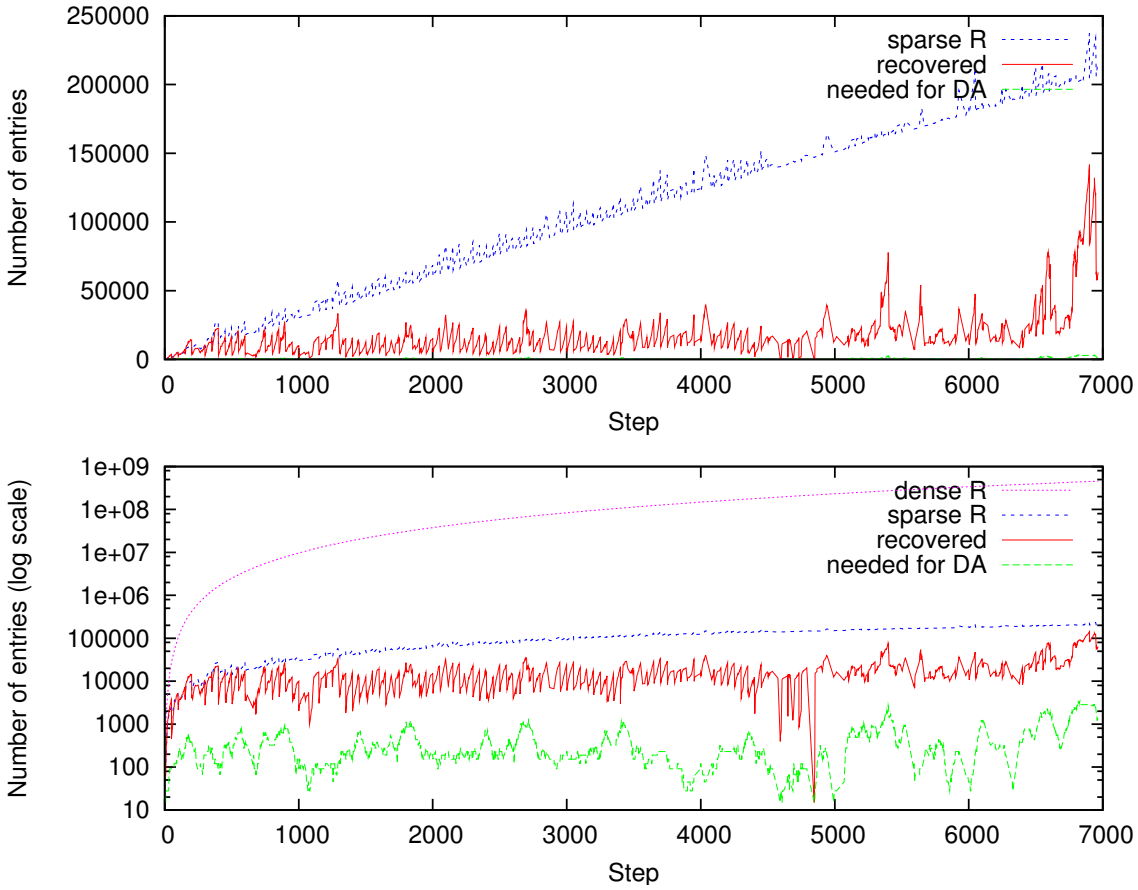


Figure 36: Number of entries of the covariance matrix that have to be recovered compared with the number actually required for data association, both in linear (top) and log scale (bottom). For reference I also show the number of entries in R and the number of entries if R was dense (log scale only). Note that a significantly lower number of entries are calculated than there are non-zero entries in R.

results for nearest neighbor data association, which does not require access to the marginal covariances. A threshold of 2 bit only removes a few measurements yielding only a slightly lower execution time. The result also shows that the selection of measurements does not add a significant overhead as the marginal covariances are already recovered for JCBB data association. For a 3 bit threshold the number of measurements is significantly lower, resulting in a significant speedup due to lower complexity of both the estimation and the marginal covariance recovery. Finally, for a 4 bit threshold too many measurements are removed and the data association fails to close loops correctly, leading to an inconsistent map as shown in Figure 35.

Table 3: Execution times for different marginal covariance recovery methods for a simulated loop. The times include updating of the factorization, solving for all variables, and performing the respective data association technique, for every step.

	Execution time		
	Overall	Avg./step	Max./step
NN	$2.03s$	$4.1ms$	$81ms$
ML conservative	$2.80s$	$5.6ms$	$95ms$
ML exact, efficient	$27.5s$	$55ms$	$304ms$
ML exact, full	$429s$	$858ms$	$3300ms$

Marginal covariance recovery is quite efficient with my dynamic algorithm. In Figure 36 I compare the number of entries of the covariance matrix that have to be recovered with the number of actually required entries for JCBB data association. The figure shows both linear (top) and log scale (bottom). The number of recovered entries is much lower than the actual number of non-zero entries in the factor matrix because my dynamic algorithm for marginal covariance recovery only calculates the entries that are actually needed. If, lets say no variables from the left half of the covariance matrix are needed, then the entries corresponding to non-zero entries in the right half of R also do not have to be calculated. Furthermore, from the remaining part of the matrix not all entries corresponding to non-zeros in the factor matrix are necessarily required, as some variables might not depend on others even if those are further to the right.

The number of entries calculated by the dynamic approach stays almost linear in this example. This really depends on the order of the variables in the R factor, as it is more expensive to obtain covariances for entries that are further to the left side of the matrix. The spikes often coincide with an increased number of non-zero entries in R, which are caused by incremental updates during loop closing events. The significant increase on the right is a combination of a denser R factor (see spikes in blue curve) with required variables being further to the left of the matrix and additionally more entries are requested (see green curve).

4.4.2 Conservative Estimates

While JCBB based on exact marginal covariances is fast enough for most cases, I still include an evaluation of the recovery of conservative estimates for completeness. All results of this section were obtained on a Pentium M 2GHz laptop.

I compare different methods for obtaining the marginal covariances for ML data association in Table 3. The results are based on a simulated environment with a 500-pose loop and 240 landmarks, with significant measurement noise added. Undetected landmarks and spurious measurements are simulated by replacing 3% of the measurements by random measurements. The nearest neighbor (NN) approach is dominated by the time needed for factorization updates and back-substitution in each step. As those same calculations are also performed for all covariance-based approaches that follow, these times are a close approximation to the overall calculation time without data association.

The results show that while my exact algorithm is much more efficient than full inversion, my conservative solution provides a high frame rate solution. In the second and third rows, the maximum likelihood (ML) approach is evaluated for my fast conservative estimate and my efficient exact solution. The conservative estimate only adds a small overhead to the NN time, which is mostly due to back-substitution to obtain the last columns of the exact covariance matrix. This is computationally the same as the back-substitution used for solving, except that it covers a number of columns equal to the dimension of a single pose instead of just one. Recovering the exact marginal covariances becomes fairly expensive in comparison as additionally the block-diagonal entries have to be recovered. My exact efficient algorithm is an order of magnitude faster compared to the direct inversion of the information matrix, even though I have used an efficient algorithm based on a sparse LDL^T matrix factorization [60]. However, this does not change the fact that all n^2 entries of the covariance matrix have to be calculated for the full inversion. Nevertheless, even my fast algorithm is expensive when calculated in every step. But as the uncertainties can never grow when new measurements are added, an alternative solution is to calculate exact values when time permits and use them to update the conservative estimates.

iSAM with conservative estimates also runs comfortably in real-time for the Victoria

Park sequence. Performing ML data association based on conservative estimates, the incremental reconstruction including solving for all variables after each new frame is added took $464s$ or 7.7 minutes, which is significantly less than the 26 minutes it took to record the data. That means that iSAM is still over 3 times faster than real-time. Obtaining the exact map and trajectory by back-substitution only every 10 steps yields a significant improvement to $270s$ or 4.5 minutes. The difference to known data association is not significant. Under known correspondences the time reduces to $351s$ or 5.9 minutes. The difference is mainly caused by the back-substitution over the last three columns to obtain the off-diagonal entries in each step that are needed for data association. The decrease is not significant because a similar back-substitution over a single column still has to be performed to solve for all variables in each step.

iSAM with conservative estimates also still performs in real-time towards the end of the trajectory, where the computations get more expensive. Even for the slow case of retrieving the full solution after every step, iSAM including data association takes on average $0.120s$ per step for the final 100 steps. These results compare favorably to the $0.22s$ needed for real-time performance. Again, this average includes a full linearization, COLAMD variable reordering step and matrix factorization, which took $1.8s$ in total.

4.5 Related Work

Common approaches to the data association problem assume known correspondences or use a maximum likelihood assignment. However, reverting to a specific correspondence assignment involves the danger of choosing a wrong correspondence, often leading to catastrophic failure of the system. In laser-based applications, correlation based approaches based on scan matching, such as presented by Gutmann and Konolige [67], are used for 2D data, and the iterative closest point (ICP) algorithm by Besl and McKay [7] for 3D data. However, these approaches do not apply to sparse landmark-based SLAM.

Another way of dealing with unknown correspondences is to generate a set of candidate assignments, and then prune inconsistent ones. This is typically done using random sample consensus (RANSAC) by Bolles and Fischler [9], a probabilistic algorithm that repeatedly

selects a minimum number of candidates needed to constrain the given problem, and determines the support from the remaining candidates. It is frequently used in SFM [6], visual SLAM [120, 21, 82] and visual odometry [110].

A more direct way of dealing with unknown correspondences is to consider multiple hypotheses. This can be based on combinatorial considerations [4], tree search [105], or lazy search that revises decision only if needed [70]. It can also be achieved in the context of particle-based representations [69, 98, 108, 38], where each particle carries its own correspondences, typically based on a per-particle maximum likelihood assignment or random sampling. The joint compatibility branch and bound algorithm by Neira and Tardos [105] has proven particularly successful [2, 131]. However, it requires access to the marginal covariances, and besides the EKF with its consistency problems, efficient direct access to these marginals was not available before our work.

An approach that avoids the combinatorial nature of the problem samples from probability distributions over correspondences [28, 31] in a batch structure from motion context. SLAM, however, requires an incremental solution. We have adapted the approach accordingly in [77], which could now be combined with iSAM.

We have presented simple data association and marginal covariance recovery for iSAM in [80, 81]. However, this dissertation contains more details and an improved dynamic programming algorithm for recovering marginal covariances. Furthermore, the joint compatibility branch and bound algorithm as well as the selection of informative measurements have not been explored before in connection with iSAM.

Chapter V

APPLICATION: VISUAL SLAM

Visual SLAM is more challenging than other SLAM problems presented in this work. First, a large number of landmarks need to be mapped, which makes EKF-based solutions unsuitable. Second, the measurement functions are highly non-linear, which is another reason against using filter-based SLAM approaches. Consequently a smoothing based solution seems appropriate. Furthermore, data association is required to close large loops, eliminating most smoothing algorithms that do not provide access to the estimation uncertainties. And finally, loop closing constraints need to be incorporated into the estimation problem with the associated global correction of the map, rendering fixed-lag smoothing approaches useless.

The visual SLAM system that I present here consists of two parts: The first part is visual odometry, which estimates the ego-motion of a moving camera by tracking features between subsequent frames. The second part is the actual SLAM system based on iSAM. It uses a subset of the input frames and therefore only uses features that were successfully tracked over multiple frames. It also uses a subset of the remaining features to integrate over time and for closing loops. I follow up with experimental results for challenging outdoor sequences.

Throughout this chapter I assume that we have rectified images with known calibration matrix

$$K = \begin{bmatrix} f_u & s & u_0 \\ & f_v & v_0 \\ & & 1 \end{bmatrix} \tag{76}$$

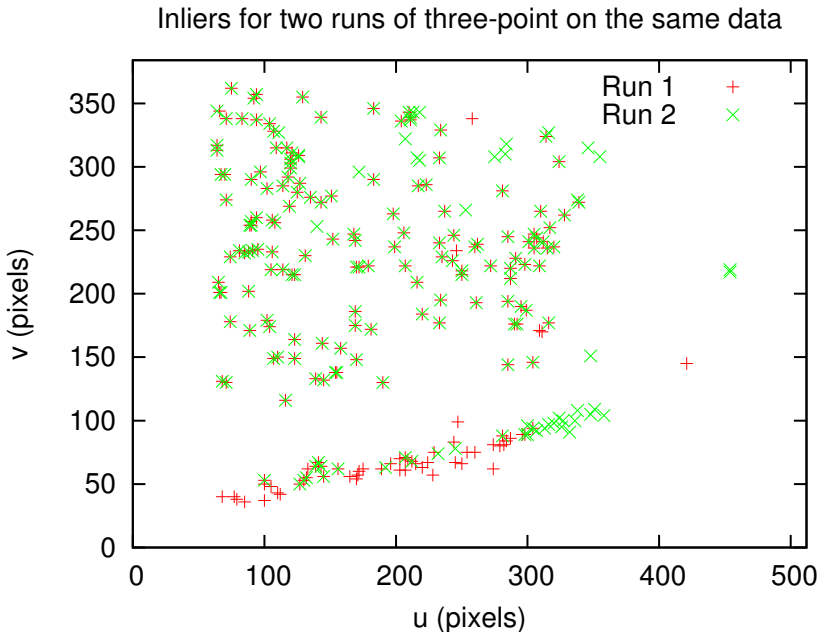
for both cameras of the stereo pair, where f_u and f_v are the focal lengths, s the skew (typically 0), and (u_0, v_0) the principal point. I define the reference camera to be the one whose pose is tracked. The other view is defined by the baseline b of the stereo pair. Camera poses are represented by a translation vector $\mathbf{t}$ and the three Euler angles yaw ϕ, pitch θ

and roll ψ, or alternatively the corresponding rotation matrix R.

5.1 Visual Odometry

I present a novel approach to visual odometry that is fast and robustly deals with nearly degenerate data. The expression "degenerate data" refers to data that is insufficient for constraining a certain estimation problem. "Nearly degenerate data" means that there are only a few data points without which the remaining data is degenerate. Almost all visual odometry work use the random sample consensus (RANSAC) algorithm [9] for robust model estimation, and are therefore susceptible to problems arising from nearly degenerate situations. It is well known that RANSAC fails when directly applied to nearly degenerate data [13, 50], but no visual odometry work addressed this so far. In visual odometry nearly degenerate data occurs for a variety of reasons, such a ground surfaces with low texture (see Figure 37(b)), bad lighting conditions that result in overexposure (which resulted in the example in Figure 37(a)), and motion blur. The consequence is that multiple runs of RANSAC on the same data yield different results, as the example in Figure 37(a) shows. While no previous visual odometry work has dealt with the problem of degeneracy, Frahm and Pollefeys [50] present a general method called QDEGSAC for robust estimation in such cases. However, their general method is not suitable here because of the hard real-time constraints of visual odometry.

An overview of our approach is shown in Figure 38. Based on extracted image features I first establish stereo correspondences as well as putative matches between two successive frames. I then separate the stereo features based on their disparity and the vehicle speed into two sets. The set with low disparity is used to recover the camera rotation with a two-point algorithm. Based on the recovered rotations, I then use the second set of features to recover the vehicle translation with a one-point algorithm. I have implemented this algorithm as well as a reference three-point algorithm on an outdoor robot with a stereo rig. I show that this algorithm is faster than the reference system, and that it successfully deals with nearly degenerate data.



(a) Two runs of standard visual odometry on the same data yield different sets of inliers in degenerate situations.

(b) Another example where a ground surface with low texture results in degenerate data.

Figure 37: Examples of nearly degenerate data based on low availability of features close to the camera, which are needed to estimate the translation. (a) An example in which the inliers for two different runs of RANSAC significantly differ, as shown by the red plus signs and green crosses. In this case, patches of strong light in otherwise dark areas of the image lead to overexposure resulting in only a small number of close features along a line. (b) A different scenario that causes a standard three-point RANSAC approach to fail because most of the structure is too distant to provide translational constraints, but a high inlier ratio can be achieved from the features in the background.

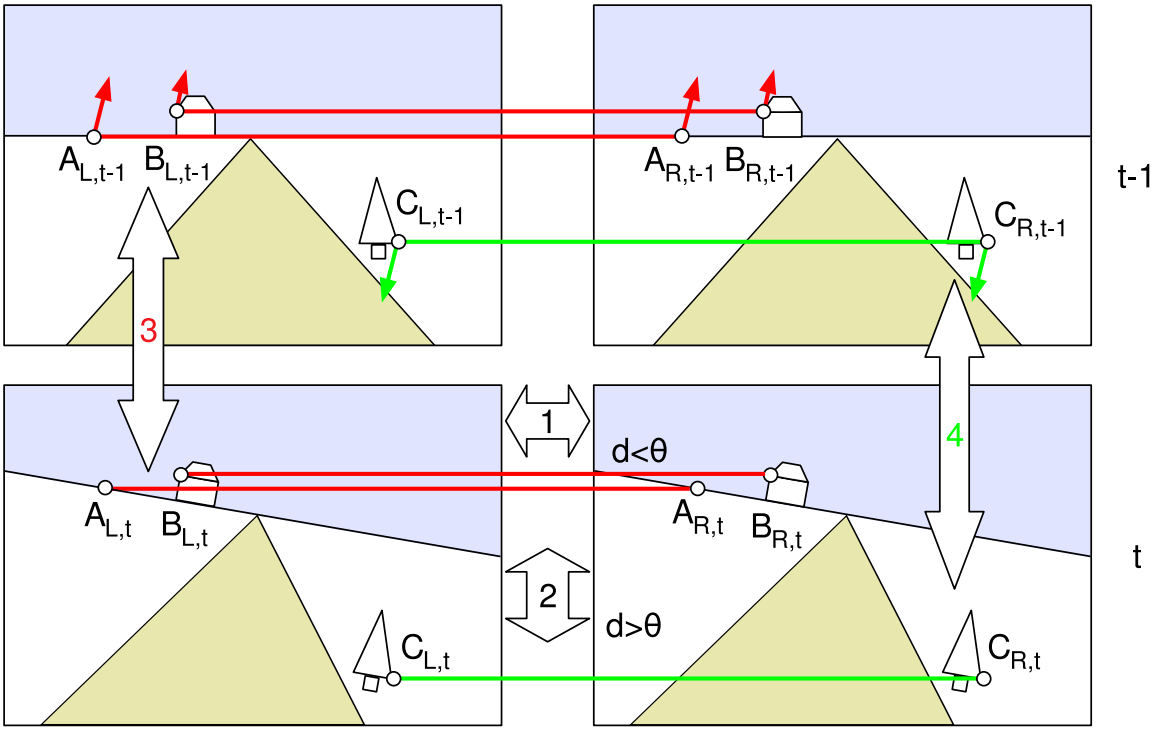


Figure 38: Our approach consists of four steps that are performed for each new frame: (1) Sparse stereo and putative matching. (2) Separate the sparse flow by disparity based on the threshold θ that adapts to the vehicle speed. (3) Recover the rotational component of the camera motion from two distant points A_t and B_t using RANSAC to robustly deal with outliers. (4) Recover the translational component of the camera motion from one close point C_t again using RANSAC.

5.1.1 Stereo Odometry by Sparse Flow Separation

As shown in Figure 38 our algorithm performs the following four steps on each new stereo pair:

1. Perform sparse stereo and putative matching.
2. Separate features based on disparity.
3. Recover rotation with two-point RANSAC.
4. Recover translation with one-point RANSAC.

Next I describe these steps in detail.

5.1.1.1 Sparse Stereo And Putative Matches

I extract features in the current frame and establish stereo correspondences between the left and right image of the stereo pair. For a feature in one image, the matching feature in the other is searched for along the same scan line, with the search region limited by a maximum disparity. As there are often multiple possible matches, appearance is typically used and the candidate with lowest difference in a small neighborhood accepted, resulting in the set of stereo features $\mathcal{F} = \{(u_i, v_i, u'_i)\}_i$, where (u, v) is the location of a feature in the reference frame and (u', v) the corresponding feature in the other frame.

Based on the stereo features $\mathcal{F}_t$ from the current frame and the features $\mathcal{F}_{t-1}$ from the previous frame I establish *putative matches*. For a feature in the previous frame, I predict its location in the current frame by creating a 3D point using disparity and projecting it back. For this reprojection we need to have a prediction of the vehicle motion, which is obtained in one of the following ways:

- Odometry: If wheel odometry or IMU are available.
- Filter: Predict camera motion based on previous motion.
- Stationary assumption: At high frame rate we obtain a small enough motion to approximate by a stationary camera.

As the predicted feature locations are not exact in any of these cases, I select the best of multiple hypotheses. I use the approximate nearest neighbors (ANN) algorithm [1] to efficiently obtain a small set of features within a fixed radius of the predicted location. The best candidate based on template matching is accepted as a putative match. I denote the set of putative matches with $\mathcal{M}$. As some putative matches will still be wrong, I use a robust estimation method below to filter out incorrect matches.

5.1.1.2 Separate Features

I separate the stereo features based on their usefulness in establishing the rotational and the translational components of the stereo odometry. The key idea is that small changes in the camera translation do not visibly influence points that are far away. While points at infinity are not influenced by translation and are therefore suitable to recover the rotation of the camera, there might only be a small number or even no such features visible due to occlusion, for example in a forest or brush environment. However, as the camera cannot translate far in the short time between two frames (0.067 seconds for my 15 frames per second system), we can also use points that have disparities somewhat larger than 0. Even if the camera translation is small, however, if a point is close enough to the camera its projection will be influenced by this translation.

I find the threshold θ on the disparity of a point for which the influence of the camera translation can be neglected. The threshold is based on the maximum allowed pixel error given by the constants Δu and Δv, for which values in the range of 0.1 to 0.5 seem reasonable. It also depends on the camera translation $\mathbf{t} = (t_x, t_y, t_z)$, which can again be based on odometry measurements, a motion filter, or a maximum value provided by physical constraints, for example of a mobile robot platform. Considering only the center pixel of the camera as an approximation, we obtain the disparity threshold

$$\theta = \max \left\{ \frac{b}{\frac{t_x}{\Delta u} - \frac{t_z}{f}}, \frac{b}{\frac{t_y}{\Delta v} - \frac{t_z}{f}} \right\}. \tag{77}$$

I separate the putative matches into the set $\mathcal{M}_{\text{rot}} = \{\mathcal{M} | \text{disparity} \leq \theta\}$ that is useful for estimating the rotation, and the set $\mathcal{M}_{\text{trans}} = \{\mathcal{M} | \text{disparity} > \theta\}$ that is useful for estimating the translation. Note that we always have enough putative matches in $\mathcal{M}_{\text{rot}}$

even if the robot is close to a view obstructing obstacle, due to physical constraints. As the robot gets close to an obstacle, its speed has to be decreased in order to avoid a collision, therefore increasing the threshold θ. On the other hand, it is possible that all putatives have disparities below the threshold θ, in particular for $\mathbf{t} = 0$. In that case we still have to use some of the close putatives for calculating the translation, as we do not know if the translational speed of the camera is exactly 0 or just very small. I therefore always use a minimum number of the closest putative matches for translation estimation, even if their disparities fall below θ.

5.1.1.3 Rotation: Two-point RANSAC

I recover the rotational component R of the motion based on the set of putative matches $\mathcal{M}_{\text{rot}}$ that are not influenced by translation. For points at infinity it is straightforward to recover rotation based on their direction. Even if points are close to the camera such that reliable depth information is available, but the camera performs a pure rotational motion, the points can be treated as being at infinity, as their depths cannot be determined from the camera motion itself. Even though the camera's translation is not necessarily 0 in our case, I have chosen the threshold θ so that the resulting putative matches $\mathcal{M}_{\text{rot}}$ can be treated as points at infinity for the purpose of rotation estimation. I will therefore take a monocular approach to rotation recovery.

While the set of putative matches contains outliers, let us for a moment assume that the matches $(\mathbf{z}^R_{i,t}, \mathbf{z}^R_{i,t-1}) \in \mathcal{M}_{\text{rot}}$ with $\mathbf{z}^R_{i,t} = (u^R_{i,t}, v^R_{i,t})$ and $\mathbf{z}^R_{i,t-1} = (u^R_{i,t-1}, v^R_{i,t-1})$ are correct and therefore correspond to the homogeneous directions (ie. $w^R_i = 0$)

$$X^R_i = \begin{bmatrix} x^R_i & y^R_i & z^R_i & 0 \end{bmatrix}^T. \tag{78}$$

Two such matches are necessary to determine the rotation of the camera for either of the following two methods:

- We estimate the rotation R together with n directions X^R_i (2 degrees of freedom, because X^R_i is homogeneous with $w^R_i = 0$), yielding $3+2n$ degrees of freedom (DOF). Each match yields 4 constraints, therefore $n = 2$ is the minimum number of correspondences needed to constrain the rotation.

- We estimate only the rotation R, by using the features from the previous time $t-1$ to obtain the direction of the points. This yields 3 DOF, with only 2 remaining constraints per match, again yielding $n = 2$.

For pure rotation, ie. $\mathbf{t} = 0$, the reprojection error E simplifies to

$$\begin{aligned} E &= \left\| \mathbf{z}_i^R - \nu^R(R, 0, \begin{bmatrix} x_i^R \\ y_i^R \\ z_i^R \end{bmatrix}) \right\|^2 \\ &= \left\| \mathbf{z}_i^R - \nu^R \left(R, \begin{bmatrix} x_i^R \\ y_i^R \\ z_i^R \end{bmatrix} \right) \right\|^2 \end{aligned} \tag{79}$$

where $(u, v) = \nu^R(R, \mathbf{t}, X)$ is the monocular projection of point X into the camera at pose $R, \mathbf{t}$. We numerically obtain an estimate for the rotation R and optionally the point directions by minimizing the nonlinear error term

$$R_t, \underbrace{\left\{ \begin{bmatrix} x_i^R & y_i^R & z_i^R \end{bmatrix}^T \right\}_i}_{\mathbf{t}_i'} = = \underset{R_t, \mathbf{t}_i'}{\operatorname{argmin}} \sum_{i, \tau \in \{t, t-1\}} \left\| \mathbf{z}_{i,\tau}^R - \nu^R \left(R_\tau, \begin{bmatrix} x_i^R \\ y_i^R \\ z_i^R \end{bmatrix} \right) \right\|^2 \tag{80}$$

where $R_{t-1} = I$ and therefore R_t the rotational component of the visual odometry. Note that we also need to enforce $\left\| \begin{bmatrix} x_i^R & y_i^R & z_i^R \end{bmatrix}^T \right\|^2 = 1$ in order to restrict the extra degree of freedom provided by the homogeneous parametrization.

While I have assumed correct matches so far, the putative matches in $\mathcal{M}_{\text{rot}}$ are in fact noisy and contain outliers. I therefore use the random sample consensus (RANSAC) algorithm [9] to robustly fit a model. The sample size is two, as two putative matches fully determine the camera rotation, as discussed above. RANSAC repeatedly samples two points from the set of putatives and finds the corresponding rotation. Other putatives are accepted as inliers if they agree with the model based on thresholding the reprojection error E from (79). Sampling continues until the correct solution is found with some fixed probability. Subsequently, a better rotation estimate is determined based on all inliers.

Finally, this improved estimate is used to identify inliers from all putatives, which are then used to calculate the final rotation estimate $\hat{R}$.

5.1.1.4 Translation: One-point RANSAC

Based on the now known camera rotation, I recover the translation from the close putative matches $\mathcal{M}_{\text{trans}}$. I denote a putative match as $\mathbf{z}_{i,t}^{\mathbf{t}}$ and the corresponding 3D points as $X_i^{\mathbf{t}}$. Each measurement imposes $2 \times 3 = 6$ constraints, i.e. $\mathbf{z}_{i,t}^{\mathbf{t}} = (u_{i,t}^{\mathbf{t}}, v_{i,t}^{\mathbf{t}}, u'^{\mathbf{t}}_{i,t})$ and $\mathbf{z}_{i,t-1}^{\mathbf{t}} = (u_{i,t-1}^{\mathbf{t}}, v_{i,t-1}^{\mathbf{t}}, u'^{\mathbf{t}}_{i,t-1})$, which now includes stereo information in contrast to Section 5.1.1.3. Intuitively we can recover the translation from a single putative match, as each of the two stereo frames defines a 3D point and the difference between the points is just the camera translation. Practically we again have two different approaches:

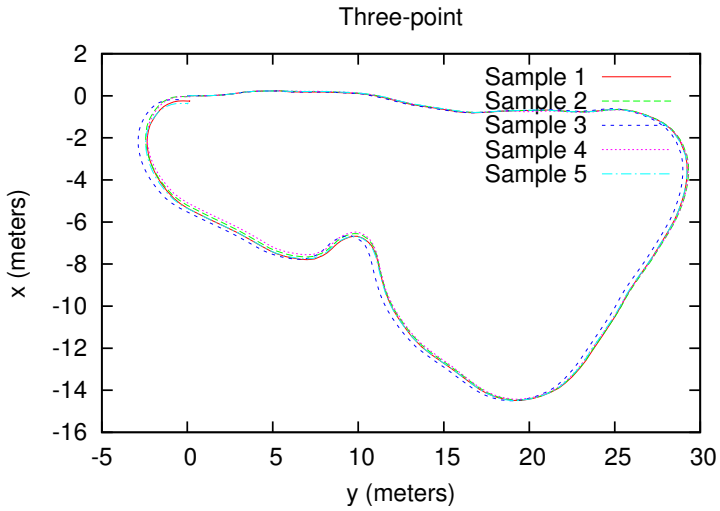
1. We estimate both the translational component $\mathbf{t}$ with 3 DOF and the 3D points $\left\{X_i^{\mathbf{t}}\right\}_i$ with 3 DOF each. Each measurement contributes 6 constraints, therefore a single match will make the system determinable.

2. We estimate only the translation $\mathbf{t}$ yielding 3 DOF, by using the previous stereo feature to generate the 3D point. Each measurement then only contributes 3 constraints, again requiring only a single match to constrain the camera translation.

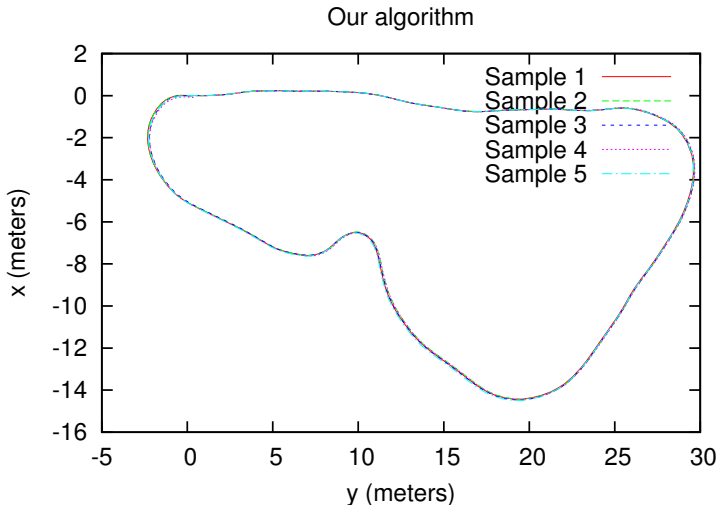
Similar to Section 5.1.1.3, the translation is recovered by optimizing over the translation and optionally also the 3D points:

$$\mathbf{t}_t, \left\{X_i^{\mathbf{t}}\right\}_i = \underset{\mathbf{t}_t, \left\{X_i^{\mathbf{t}}\right\}_i}{\operatorname{argmin}} \sum_{i,\tau \in \{t,t-1\}} \left\| \mathbf{z}_{i,\tau}^{\mathbf{t}} - \nu^{\mathbf{t}} \left(\hat{R}_\tau, \mathbf{t}_\tau, X_i^{\mathbf{t}} \right) \right\|^2 \tag{81}$$

where $(u, v, u') = \nu^{\mathbf{t}}(R, t, X)$ is the stereo projection function, and I choose $R_{t-1}, \mathbf{t}_{t-1}$ to be a camera at the origin, and $R_t = \hat{R}$ is the rotation recovered by the two-point algorithm. Consequently, $\mathbf{t}_t$ is the translation of the camera that we are interest in. Again, I use RANSAC to robustly deal with outliers, where each sample defines a translation according to (81) and the final model is also determined by (81) using all inliers.



(a) Multiple runs yield different trajectories for three-point.



(b) Our algorithm returns nearly stable results.

(c) Camera images.

Figure 39: Trajectories from multiple runs on the San Antonio dataset of (a) the reference three-point implementation and (b) our algorithm. Some images of the environment are shown in (c). It is apparent that the three-point algorithm returns inconsistent trajectories, while our algorithm provides almost the same result in each run.

5.1.2 Experiments and Discussion

I compare both efficiency and accuracy of our algorithm with a reference three-point algorithm based on multiple datasets from an outdoor robot equipped with a stereo camera. For our reference three-point algorithm I again use nonlinear optimization, but this time to simultaneously recover rotation, translation and optionally the 3D points by

$$R, \mathbf{t}, \{X_i\}_i = \underset{R, \mathbf{t}, \{X_i^{\mathbf{t}}\}_i}{\operatorname{argmin}} \sum_i \left\| \mathbf{z}_i - \nu \left(K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} X_i \right) \right\|^2 \tag{82}$$

where I use the measurements $\mathbf{z}_i$ and corresponding points X_i from all putative matches $\mathcal{M}_{rot} \cup \mathcal{M}_{trans}$. The implementation is shared between the two algorithms to make comparison as fair as possible.

In my implementation, I use the Harris corner detector [71] to obtain features in the reference frame and use dense stereo information to establish stereo correspondences for those features. Based on images with a resolution of 512×384 pixels, I extract a maximum of 600 features per image, by detecting a maximum of 50 features in each region of a 3×4 grid to ensure that the features are well spread out over the image. I use the Levenberg-Marquardt algorithm for all nonlinear optimizations. For RANSAC, I use a 99% confidence, with a maximum number of 1000 iterations to guarantee real-time performance even for frames where no consistent model can be found. I do not estimate 3D points and directions, but instead generate them based on the previous frame to speed up the nonlinear optimization. For sparse stereo flow separation I use the constant threshold of $\theta = 8$ pixels determined from (77) with a baseline of $0.12m$ and based on the maximum speed of the mobile robot platform. Timing results are obtained on a Core 2 Duo 2.2GHz laptop computer.

5.1.3 Degenerate Data and Robustness

I show that the algorithm deals well with nearly degenerate data, which is often encountered in outdoor visual odometry settings, but has not been addressed by any visual odometry work so far. Figure 39(a) shows how degeneracy influences the recovered trajectory by returning inconsistent results. This San Antonio sequence consists of putative matches (without images) recorded at 15 frames per second during a live demo of my work. As

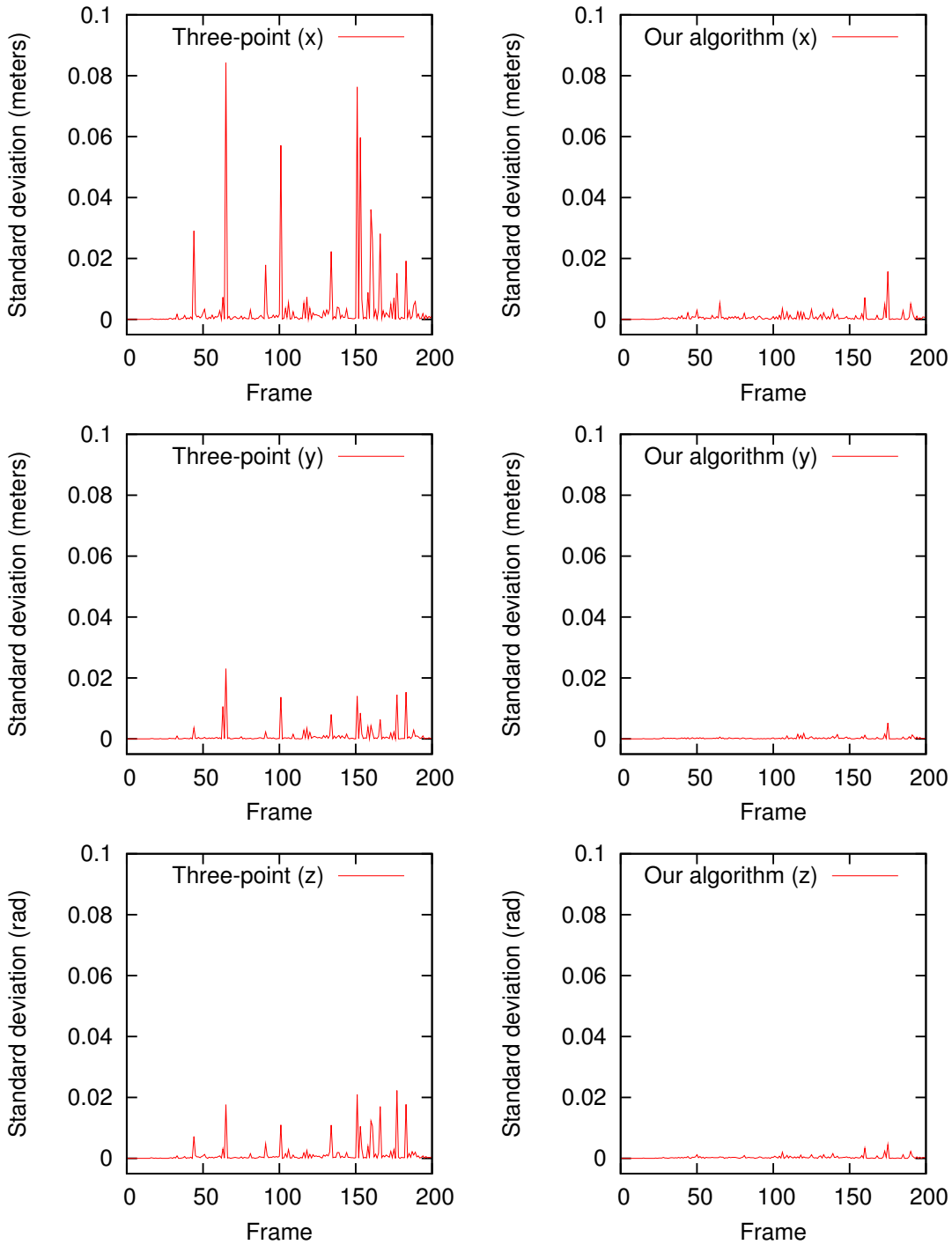
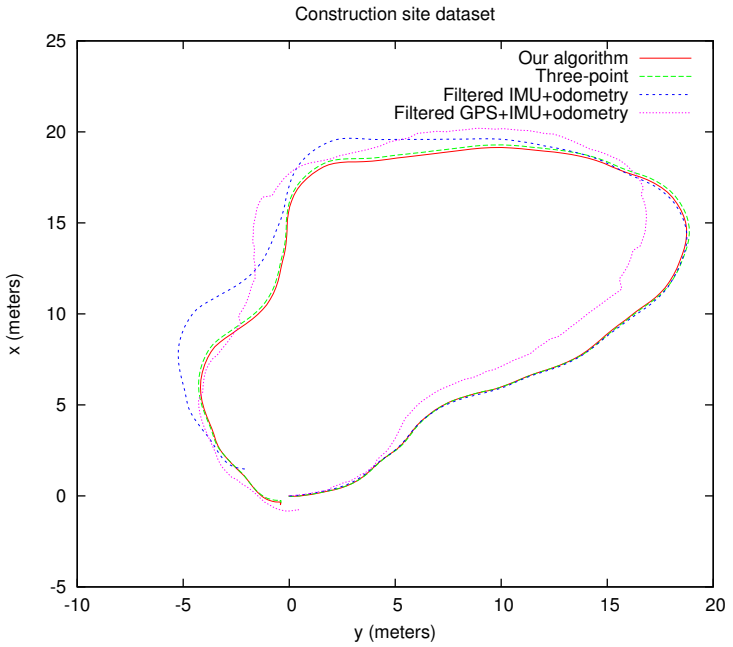


Figure 40: Standard deviations for the x, y and z components of 15 runs over the first 200 frames of the San Antonio sequence for both the reference three-point algorithm (left column) and our algorithm (right column). While the three-point algorithm shows significant variation on some frames, our algorithm yields almost stable results, in agreement with Figure 39.

can be seen, each run on the same data yields a different trajectory. Figure 39(b) shows that our algorithm returns nearly consistent results over multiple runs. Figure 40 analyzes the problem in more detail for the first 200 frames of this sequence. While our algorithm suffers only minor variations (right column), the three-point algorithm suffers significant variations for some frames (left column). The variations are mainly in the forward (x) direction that is most difficult to constrain, but also in the other two directions. Figure 37(b) shows an example of two different sets of inliers for one frame, which results in significantly different translation estimates. As discussed earlier, the problem is that RANSAC quickly finds a large number of inliers, most of which constrain the rotation, while only a few matches provide a possibly wrong translation estimate. Based on the large number of inliers, RANSAC severely underestimates the number of samples needed to achieve a certain confidence in the results. Our algorithm instead splits the problem and therefore avoids the degeneracy as is apparent from the results presented here.

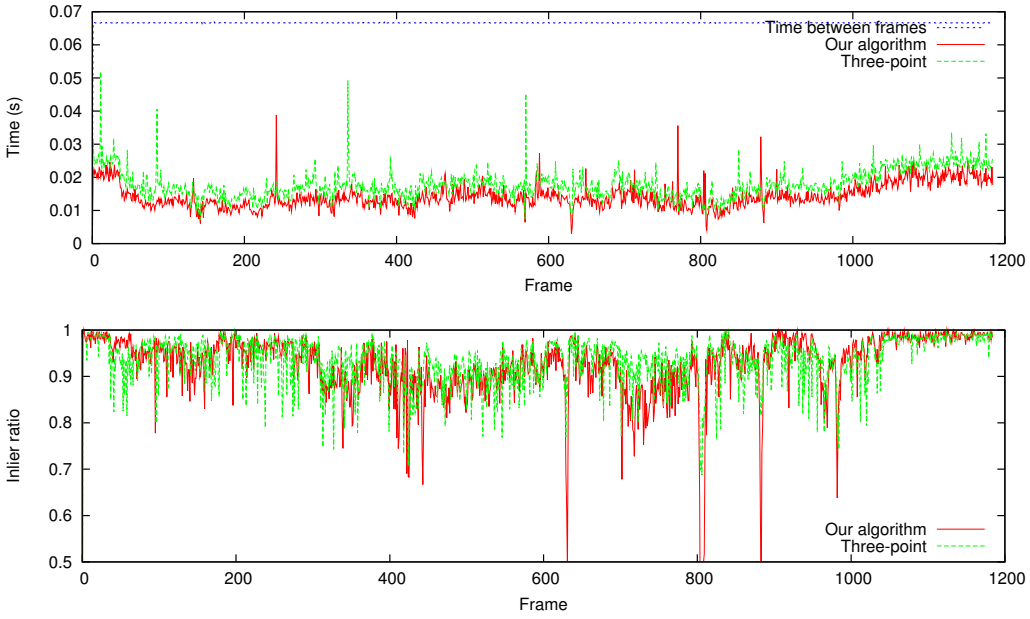
5.1.4 Speed and Accuracy

I show that the algorithm performs faster than the standard three-point algorithm, while producing at least comparable results. Figure 41 shows results from my construction site dataset, for which the images have been recorded at 15 frames per second. Figure 41(a) shows the recovered robot trajectory together with the combination of wheel odometry and inertial measurement unit (IMU), as well as combined with GPS. Figure 41(b) shows sample images from this challenging sequence, which features vastly changing lighting conditions. In Figure 41(c) I show comparisons for the execution times and inlier ratios. Our algorithm performs faster than the three-point algorithm, even though we perform RANSAC twice, once for rotation and once for translation. The faster execution is explained by the smaller sample size for each case as compared to three-point. Therefore it is more likely to randomly select a good sample and consequently RANSAC needs less iterations, assuming that both have the same inlier ratio. The inlier ratios for the two-point component of our algorithm as well as for the three-point algorithm are very similar as shown in the bottom diagram.



(a) Trajectories based on multiple algorithms and sensors.

(b) Camera images.



(c) Timing and inlier ratios comparing three-point with our algorithm.

Figure 41: Results from my construction site dataset that compare our algorithm with the reference three-point algorithm, showing increased efficiency under comparable quality. (a) The start and end points of the actual trajectory are at the origin. (b) The robot was driven over grass and gravel under difficult lighting conditions. (c) Top: Time required for RANSAC for our approach (one-point and two-point together) and the reference three-point implementation. Bottom: Inlier ratios for the two-point component of our algorithm and the reference three-point algorithm. The inlier ratio is comparable for all, but RANSAC converges faster for a smaller sample size, and therefore our algorithm is faster.

5.2 Visual SLAM

To perform visual SLAM, I use the features tracked by our visual odometry system in combination with iSAM and the JCBB algorithm for data association. In contrast to other applications in this dissertation, it is too expensive to apply iSAM and JCBB directly to the full high frame rate input data because of the large number of feature points tracked by visual odometry. Instead we have to be intelligent about which features to include in the process, which I describe in detail below.

5.2.1 Feature Selection and Landmark Creation

Due to the large number of features provided at a high frame rate by visual odometry we have to select which ones to use for visual SLAM. Visual odometry tracks an average of about 300 features per frame, 15 times per second. First, there is no gain from performing SLAM at 15 frames per second, in fact, it might be rather difficult to do so. Instead, we incorporate data only from every m^{th} frame, where typically $m = 7$. Second, as we are interested in feature tracks obtained from visual odometry, we extend those tracks across the frames we omit and only use features that were successfully tracked over those m frames.

To create new landmarks we use some of the features that have not been matched in the data association described below. In particular, for the first frame, all features are candidates for new landmarks. In order to provide good constraints on the camera pose we force the measurements to be spread out over the complete image. For that purpose we tile the image (here in a 3×4 grid) and for each region we check if we already have a measurement based on data association. If not, we select from the remaining features of this region the one with highest Harris response in order to create a new landmark. Using the highest Harris response makes it more likely that the corresponding feature is also found in the next frame.

5.2.2 Data Association

We use the JCBB algorithm to obtain local data association with respect to recent frames. Instead of using the tracks provided by visual odometry, we perform data association from

scratch. To keep computations simple, we only match against landmarks that have been observed in at least one of the previous n frames, and we are intelligent about which features to insert as described below.

There are multiple reasons for performing data association from scratch. First, the feature tracks obtained from visual odometry by connecting frame by frame tracks across multiple frames might be wrong. While RANSAC in visual odometry ensures that they confirm with the geometry on a frame by frame basis, wrong correspondences can still be generated. Connecting tracks with one wrong correspondence between can lead to a track that does provide wrong geometric constraints, therefore influencing the quality of the result. However, we still make use of the information provided by visual odometry by adding all features predicted by VO based on the landmarks observed in the previous frame.

A second reason for performing data association from scratch is that we can extend feature tracks that visual odometry was not able to continuously provide. Reasons for failing to track a feature for one or a few frames include noise and occlusion. Again we add features intelligently by adding the closest feature for each reprojection of a landmark seen in the last n frames. Taking the closest feature is sufficient as locally the pose estimates provided by visual odometry are of a high quality.

JCBB combines confirming tracks provided by visual odometry with extending tracks that visual odometry failed to provide. In particular it uses the correlation between landmarks to make an informed decision. While this improves the quality of the SLAM estimate locally, it does not yet allow to close large loops.

5.2.3 Closing Large Loops

Because JCBB cannot handle a large number of landmarks and features, we have to be careful when closing large loops. We cannot simply match all features of the current frame to all potentially visible landmarks in the map. In particular, we have to consider the uncertainty of the camera and also include landmarks that are not necessarily visible in the image based on the mean pose estimate. Therefore I have chosen to perform loop closing in an analogous way to the incremental data association. I use the current image features and

Figure 42: The DARPA LAGR mobile robot platform with two front-facing stereo camera rigs, a GPS receiver as well as wheel encoders and an inertial measurement unit (IMU).

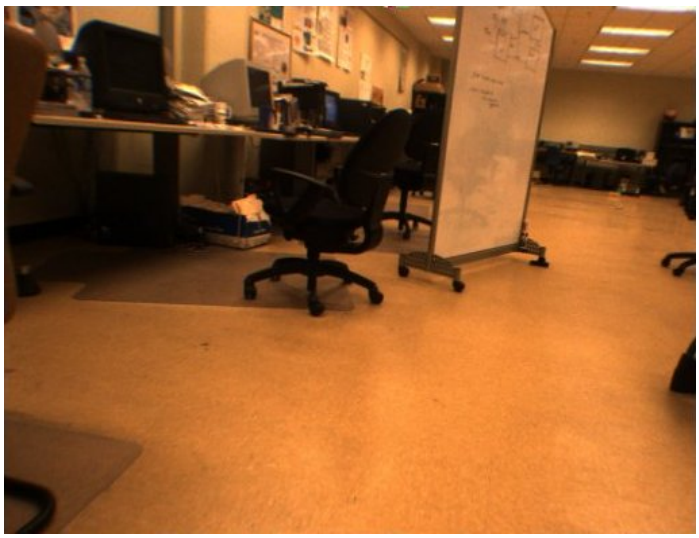
the landmarks seen from a specific frame in the past. That frame can be determined based on the motion uncertainty described in Section 4.1.4. When another pose of an earlier part of the trajectory comes into this uncertainty region, we test for loop closing against the landmarks that were observed from that frame, using JCBB to provide robust matching.

5.3 Experiments and Results

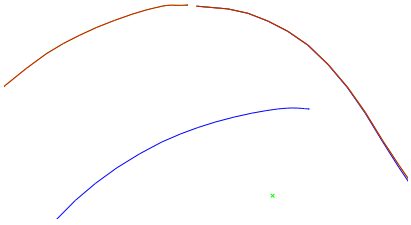
I have evaluated my algorithm on the DARPA LAGR platform shown in Figure 42. The robot features two forward-facing stereo cameras, one of which I use in these experiments. Visual odometry was run online on the robot (Core Duo 2GHz, multithreaded). The SLAM results are obtained on a Core 2 Duo 2.2GHz laptop computer, using only one core (iSAM is single threaded).

5.3.1 Incremental Data Association

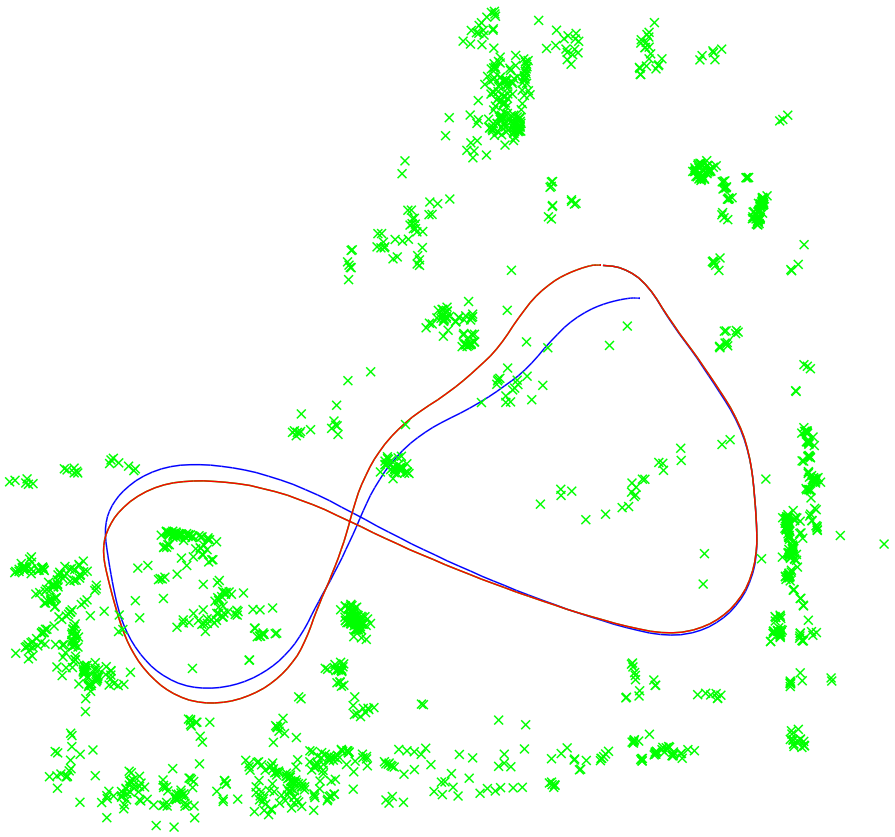
I present results for a large indoor lab environment in Figure 43. The size of the L-shaped room is about $20m$ by $20m$, with a robot trajectory length of $46m$. I use every fourth frame processed by visual odometry, resulting in 220 frames at close to $4Hz$. Visual SLAM with a complete solution in every step took about $23s$, corresponding to an average of $0.10s$ per



(a) One input image as seen by the robot.

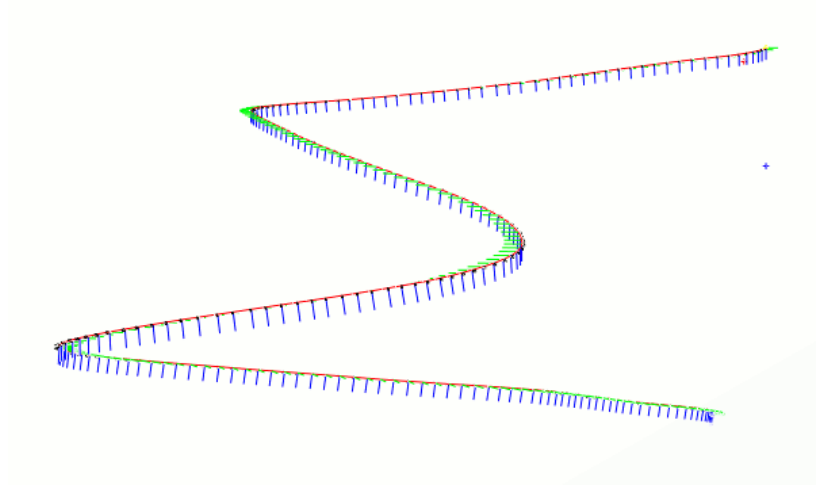


(b) Enlarged point of loop closing. Visual odometry (shown in blue) was not able to close the loop.

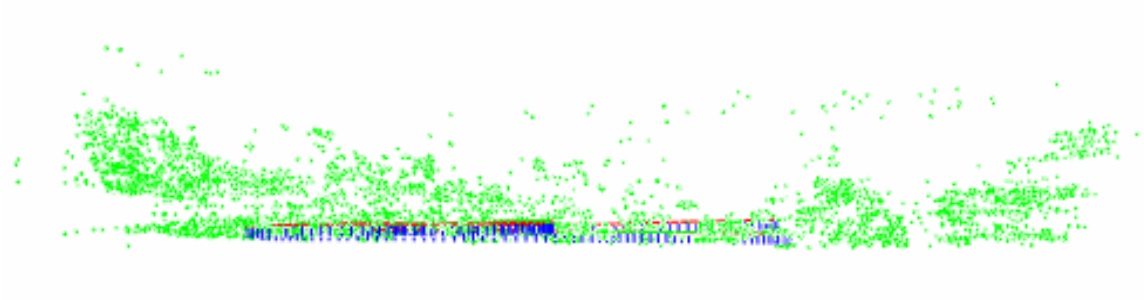


(c) Robot trajectory and landmarks after performing visual SLAM. The loop is successfully closed, and the outline of the L-shaped room is clearly visible.

Figure 43: Incremental data association for track joining increases the accuracy as shown by the results on this large indoor lab environment. Visual odometry is shown in blue, the trajectory from iSAM in red and landmarks appear green.



(a) Trajectory as recorded by the robot. There is a significant drift in height from the IMU.



(b) After optimization, the trajectory is now clearly planar as expected.

Figure 44: Side view of the trajectory for the indoor environment from Figure 43.

frame. The map contains 1643 landmarks based on 4499 image measurements.

Visual SLAM without loop closing already provides an accurate solution as can be seen in Figure 43(c) due to longer range constraints. Note that the resulting elevation is also correct as shown in Figure 44(b). In contrast, Figure 44(a) shows the trajectory as recorded by the robot, which suffers from significant drift of the inertial measurement unit (IMU).

5.3.2 Loop Closing

I evaluate my visual SLAM on data recorded during the final DARPA LAGR program demo in San Antonio, Texas. Another dataset following the same path is shown in Figure 39, with a trajectory length of about $80m$. This dataset was chosen for this experiment because of its bad quality. Note that visual odometry was running in real-time on the robot (Core Duo 2GHz) and therefore only the resulting feature tracks are available. In particular, visual odometry failed over a number of frames, and only vehicle odometry is available in those places. Even with extended tracks for the remaining frames the loop remains far from closed, see Figure 45(top).

Despite the large error at the end of the loop, JCBB succeeds in closing the large loop, with the result shown in Figure 45(bottom). Note that I have hard coded at which frame JCBB is used to close the loop. After the loop closing, two batch steps are needed because of the high nonlinearity of the measurement function to arrive at the trajectory as shown. Nevertheless, the results show how iSAM allows for data association even under such difficult circumstances with non-descriptive features. And it shows how iSAM successfully deals with a large loop closure under highly nonlinear measurement models.

5.3.3 Live Demo

While for previously presented results iSAM was run on recorded data, I have also shown iSAM performing live on the DARPA LAGR platform. The trajectories for one of the live demo runs are shown in Figure 46. The run took $121s$ and covered a trajectory of $70m$ length, partially on black top and partially on a sloping surface with tall grass. Note that this includes the robot being stuck with significant wheel slippage while backing up. The internal vehicle state (IMU+odometry) drifted and ended up meters away from the

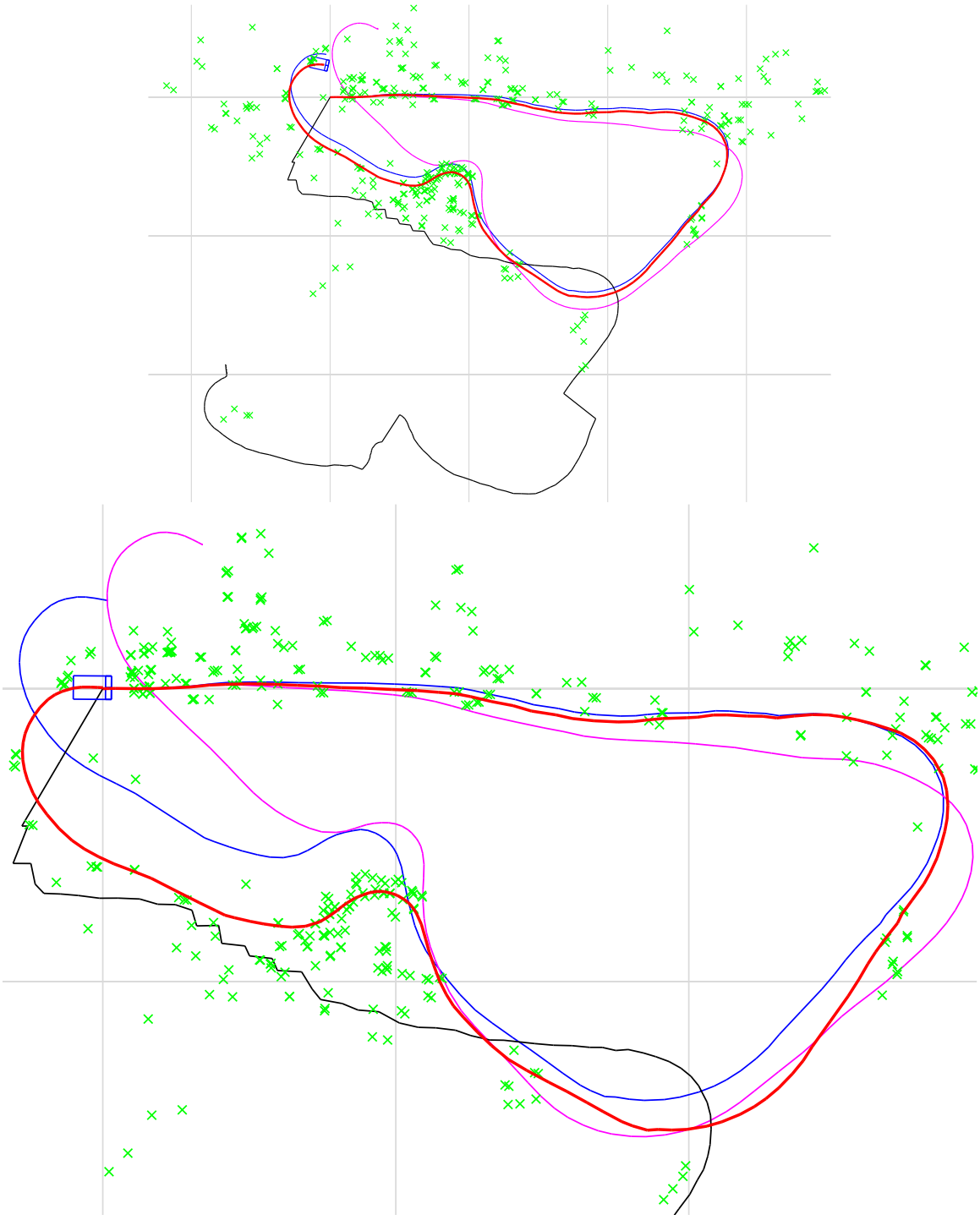


Figure 45: San Antonio sequence just before (top) and after (bottom) loop closing using JCBB. The vehicle internal trajectory estimate with GPS is shown in black, without GPS in magenta, the trajectory according to my visual odometry in blue and finally the result of my visual SLAM algorithm in red. The robot is shown as rectangular outline and the gray lines represent a grid with $10m$ spacing. Note that GPS drifted before the vehicle started to move.

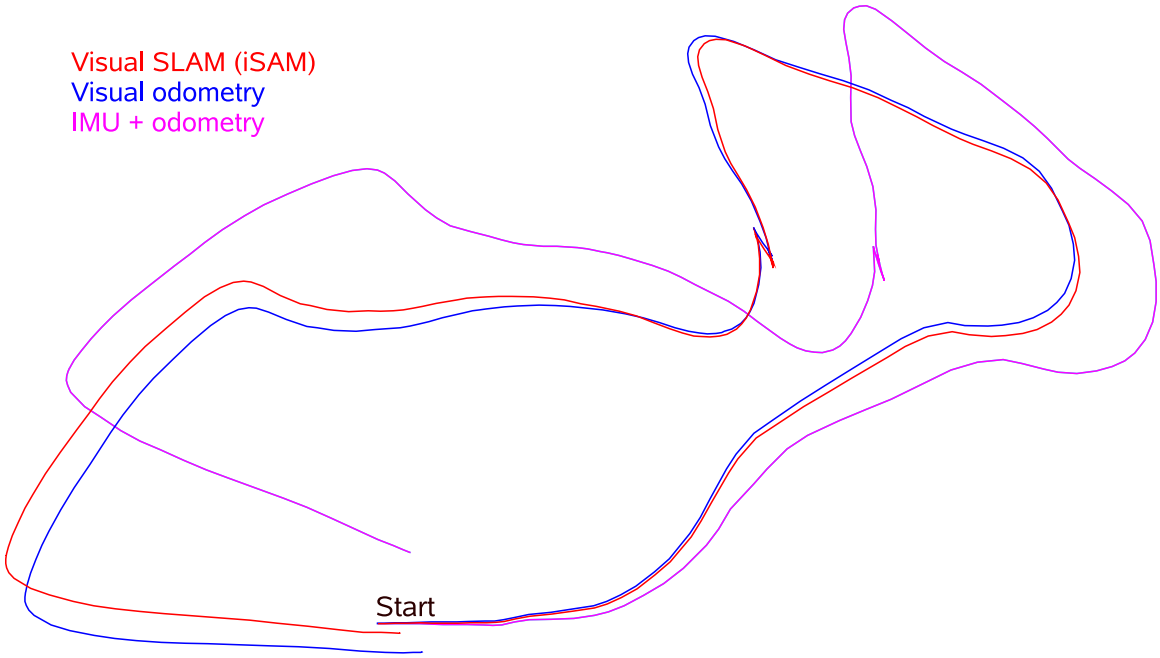


Figure 46: Robot trajectories from live demo of iSAM on the LAGR platform. The trajectory length is about $70m$ and includes a place where the robot got stuck and had to back up with significant wheel slippage. Note that the vehicle's internal state (IMU+odometry) is significantly off, while visual odometry provided a much better estimate and iSAM closed the loop.

goal, which was identical with the starting point. Visual odometry performed much better, but was still off. iSAM finally closed the loop by matching against landmarks seen at the beginning of the trajectory. However, the robot was stopped before the solution fully converged.

5.3.4 3D Maps

While so far we have focused on a sparse map that only serves for finding the robot trajectory and closing loops, we now present some results of dense 3D maps. Figure 47 shows a batch reconstruction of points corresponding to Harris features tracked by visual odometry. Figure 48 shows a much faster reconstruction based on dense stereo based on known robot poses after application of visual SLAM.

Figure 47: Batch 3D point cloud reconstruction based on the feature tracks provided by visual odometry. Every frame and all successfully tracked features from visual odometry where used, resulting in longer computation times. Each sphere corresponds to one feature, which is colored based on the corresponding pixels in the input images. All spheres have the same size, so that their reprojected size is due to perspective effects. A robot camera image from a similar view point is shown for reference.

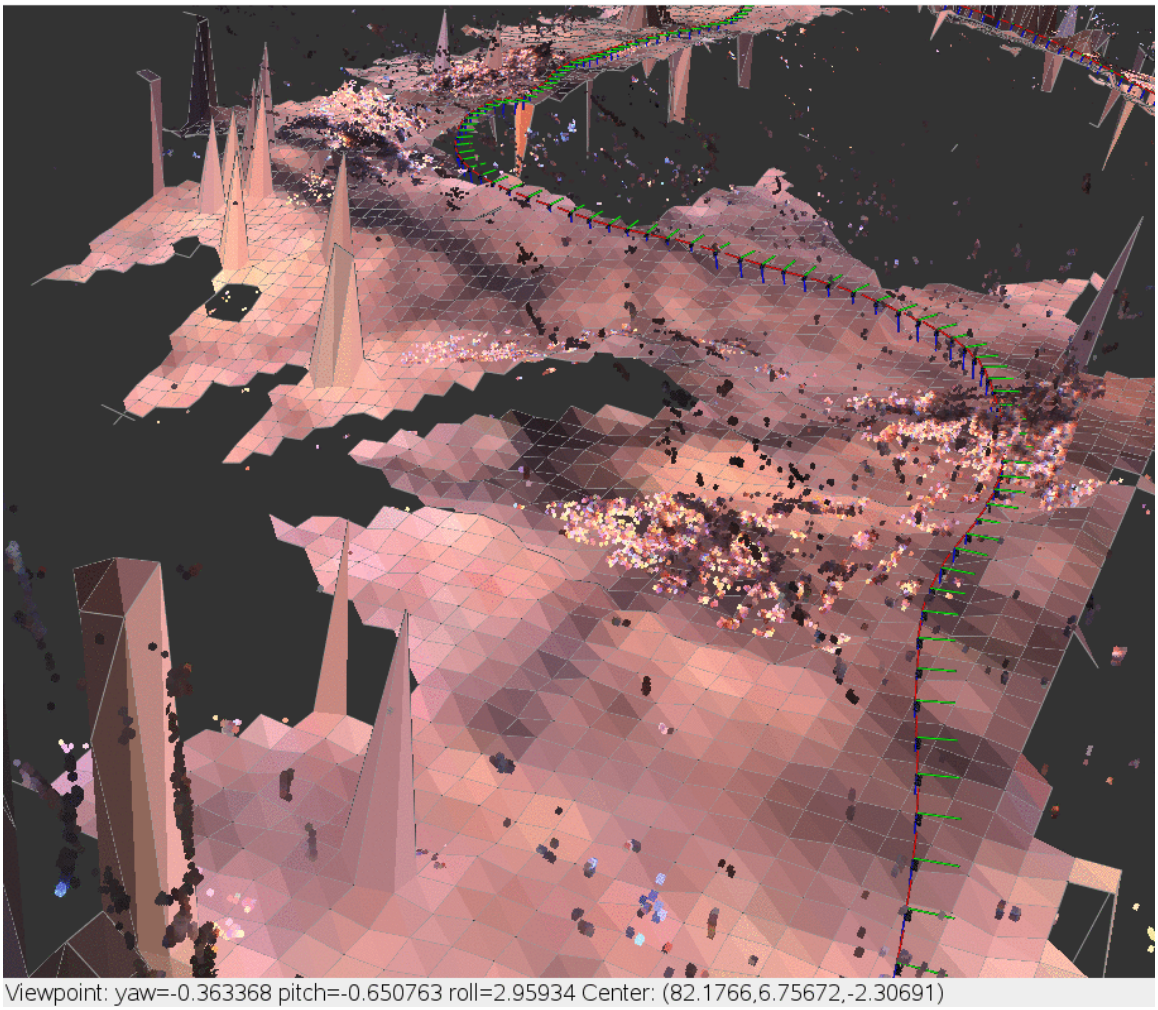


Figure 48: 3D ground surface reconstruction from dense stereo data, based on pose information from visual odometry. All pixels with available stereo information were used from frames selected at $3m$ intervals along the trajectory. A ground surface was fit by imposing a regular grid and taking the median of the height for each cell with a sufficient number of points. Only points not well described by the ground surface remain in the view. The result for this long sequence took only a few seconds to calculate.

5.4 Related Work

Visual odometry and visual SLAM are very active fields. I address some of the most important work here, but this overview is by no means complete.

5.4.1 Visual Odometry

Visual odometry is at its heart a camera pose estimation [73], and has seen considerable renewed attention in recent years. Olson [112] uses visual odometry and incorporates an absolute orientation sensor to prevent drift over time. Nister *et al.* [110] present a real-time system using a three-point algorithm, which works in both monocular and stereo settings. Levin and Szeliski [92] use loopy belief propagation to calculate visual odometry based on map correlation in an off-line system. Some systems [12, 15] also use omni-directional sensors to increase the field of view. Ni and Dellaert [106] present an image-based approach that has high computational requirements and is not suitable for high frame rates. Large-scale visual odometry in challenging outdoor environments is presented by Konolige *et al.* [88, 87, 89] and Nister *et al.* [111], but the problem of degenerate data is not addressed. Recent work by Zhu *et al.* [138] remembers landmarks to improve the accuracy of visual odometry.

While accuracy of the presented system can further be improved by applying fixed-lag smoothing [111], by using key frames to prevent drift [89], and by using landmarks to provide longer range constraints [138], in our case visual SLAM takes care of the long range consistency. Note that fixed-lag smoothing does not solve the problem of degeneracy, as only frame-to-frame correspondences are used. As those frame-to-frame correspondences are affected by a failure of RANSAC to find correct matches that constrain the translation, the same error will also be present after fixed-lag smoothing as no new matches will be added by fixed-lag smoothing.

5.4.2 Visual SLAM

One of the first visual SLAM systems is described by Harris and Pike [72], using one EKF per feature. More advanced systems were based on monocular [104], stereo [135, 103], and omnidirectional vision [55].

Visual SLAM is deeply related to bundle adjustment, a structure from motion approach in computer science. Triggs *et al.* [134] is probably the most complete seminal work on bundle adjustment, and covers a large range of important aspects, many of which are also useful for visual SLAM. The work provides insights into how to correctly implement bundle adjustment. It includes an overview of different numerical optimization methods for solving the underlying estimation problem. Matrix decompositions and incremental techniques are also addressed, as well as error modeling, preconditioning, and the problem of gauge freedoms.

More recently, several interesting lines of work can be identified. One thread of work started with Se *et al.* [120, 121] that uses SIFT features in combination with a trinocular sensor. The estimation process is based on the Kalman filter, and real-time loop closing results are shown, assuming loop-closure with the first submap. In [122, 39, 124], this work was recently combined with FastSLAM to apply to larger scale environments. Sim *et al.* [123] provide a general framework for vision-based SLAM based on Rao-Blackwellized particle filters. Experimental results compare the algorithm's behavior for the cases of visual odometry only for monocular as well as stereo sensors.

Another line of work was started by Davison, using high frames rather than complex feature descriptors. Davison [20] presents a real-time 3D SLAM system that starts from a known state, tracking known features to avoid the initialization problem. This work is based on their previous active vision work [24, 26, 25]. In later work, Davison [21] extends the system to a desk-like environment, running at 30Hz. The size of the environment is restricted by the number of features that can be tracked by the EKF in real-time, stated to be around 100. As matching is performed against structure, the search region for matching is restricted based on a motion model. Davison *et al.* [23] compares performance of the system with a wide-angle camera version. Recent work by Davison [22] analyzes information-theoretic guided search for Gaussian measurement and model uncertainties as a substitute for RANSAC. Their system has recently been applied to humanoids in [129]. Montiel *et al.* [101] presents an alternative feature initialization that allows points at arbitrary depths. Smith *et al.* [125] claims to be first monocular real-time SLAM using lines, and shows

preliminary results. Davison *et al.* [27] provides a good summary of their previous work.

Davison's work was also combined with FastSLAM by Eade and Drummond [37] for use with larger environments. Still, experiments with only 200 landmarks are shown, and a questionable analysis of the required number of particles is presented. vSLAM by Karlsson *et al.* [83] is another FastSLAM based approach based on monocular vision and is available as commercial software. However, map learning is a lengthy process, and the size of the environment is restricted to a small room. Marks *et al.* [95] presents another approach to visual SLAM based on FastSLAM that uses grid-based mapping of height variances for outdoor environments.

Another line of work focuses on reconstructing the trajectory only [110, 88], which became known as visual odometry. Nister *et al.* [110] estimates the motion of a single camera or stereo head incrementally. A robust estimation scheme is presented that is based on preemptive RANSAC [109] in connection with the 5-point and 3-point algorithms. Results are shown for outdoor sequences, comparing to DGPS and INS. Mapping can be performed by using the trajectory information to plot down the map. However, since landmarks are not part of the estimation problem, it is not suitable for loop closing. However, the incremental accuracy achieved is remarkable [111]. Konolige *et al.* [88] present an overview and results of SRIs software for the DARPA LAGR program, including visual odometry and mapping.

Engels *et al.* [40] propose that fixed-lag smoothing using bundle-adjustment is fast enough for real-time applications if implemented correctly. The resulting constant size linear system is completely solved by Cholesky factorization in each step. We have presented a batch mode bundle adjustment approach in [78, 32] that works at low frame-rate by making use of the superior constraints provided by a multi-camera rig.

Recovering the complete robot trajectory without landmarks leads to an exactly sparse information matrix as observed by Eustice *et al.* [42, 43]. Landmarks are omitted by converting the image measurements into constraints between poses. This is very similar to my work if applied to a pose-only setting, but cannot recover the exact marginal covariances. An iterative process is proposed instead that keeps track of conservative estimates.

Clemente *et al.* [14] presents an interesting approach to visual SLAM that uses the

simplified version of JCBB (see Section 4.1.3.3) in combination with an EKF. I am using the full JCBB algorithm for data association in this chapter. Additionally, my approach allows recovery of the exact values from a smoothing approach that avoids the problems inherent to the EKF.

Chapter VI

DISCUSSION

I have presented iSAM, a fast incremental solution to the SLAM problem that updates a factorization of the sparse smoothing information matrix. By employing smoothing, ie. keeping all pose variables in the estimation problem instead of performing marginalization, we obtain a naturally sparse information matrix. My approach is based on a direct equation solver using QR matrix factorization, which provides advantages over iterative solvers. Most importantly, iSAM allows for efficient access to the exact marginal covariances. I have presented applications of iSAM to well-known data association techniques such as joint compatibility, and I have shown how to use iSAM to determine how much information a measurement provides about the state estimate. I have evaluated iSAM for a variety of SLAM problems, on both simulated and real data, for different environments and sensors, in landmark and pose-only settings.

6.1 Thesis

The thesis statement presented in Chapter 1 can now be restated with all the claims defended through experimental results:

> *Incremental smoothing and mapping (iSAM) provides a superior alternative to previous SLAM approaches that is exact, is efficient, supports data association by providing access to the exact marginal covariances, and is generally applicable to a wide range of SLAM problems.*

In particular, the four claims contained in my thesis are defended by the results provided in this dissertation:

1. *iSAM is exact:* I have shown that iSAM provides results that are very close to the maximum likelihood solution. iSAM does not perform any approximations, leaving

linearization errors as the only source of temporary inaccuracy. iSAM is based on a nonlinear equation solver that finds the maximum likelihood solution even for nonlinear functions in multiple iterations, as long as it is started from a good initial estimate, which is typically available in incremental SLAM. However, iSAM only performs one iteration per step, and performs relinearization less frequently. Nonetheless, my experiments show that most of the time iSAM provides results that are very close to the least-squares solutions. The intuitive explanation is that variable estimates get refined over time, which requires only a few steps because of the quadratic convergence of the direct solver. Therefore only the most recently added variables are not fully converged yet.

2. *iSAM is efficient:* I have shown that iSAM compares favorably with state of the art SLAM algorithms. Incrementally adding new measurements into the factor matrix is a constant time operation for exploration tasks. As SLAM frequently explores previously unvisited areas (otherwise we could just do localization!), it operates at low cost most of the time. When loops are closed the variable reordering helps to keep costs low, even though some batch steps are necessary. Overall, my experimental results show that iSAM performs much faster on standard SLAM datasets than required for real-time application, and also compares well with other state of the art solutions to SLAM.

3. *iSAM supports data association:* I have shown how iSAM provides efficient access to the exact marginal covariances needed by state of the art data association techniques. In particular, I have presented results using the joint compatibility branch and bound algorithm for both laser-based and camera-based datasets. I have further shown how iSAM provides a measure of the information theoretic value of individual measurements. For both cases, my dynamic programming approach for obtaining the required marginal covariances provides an efficient solution based on the sparse factor matrix.

4. *iSAM is general:* I have shown that iSAM works for a range of SLAM applications. I have shown indoor and outdoor applications, using laser-range sensors and cameras.

I have solved for trajectories based on pose constraints derived from laser-based scan matching. And I have solved for the location of landmarks, where landmarks were obtained from Harris features in images, or from laser data by a tree detector. For quantitative evaluations I have also made use of simulated data in order to have ground truth information available.

6.2 Assumptions

The first assumption I make is that the measurement noise is Gaussian or at least can be well approximated by a Gaussian distribution. Assuming Gaussian measurement noise is common and reasonable for SLAM algorithms. A potential disadvantage is that outliers might have a large influence on the final estimation result. One way to deal with this problem is to identify outliers and remove them. However, identifying outliers is not always easy, as a measurement that for example yields a large residual is not necessarily an outlier, but can instead represent a particularly informative measurement. An alternative to Gaussian distributions is to use heavy tailed error functions as occasionally done in the structure from motion community; however, this has not been applied to iSAM so far. When making a Gaussian assumption it is also important to make the correct choice in parametrization, for example when dealing with depth information from stereo in visual SLAM. It is apparent that the noise distribution on the depth by no means follows a Gaussian distribution. However, when one chooses the original image measurements instead of the depth to represent the disparity then the Gaussian assumption becomes reasonable once again.

The second assumption I make is that the error function to be optimized is locally convex. Because the SLAM error function is all but convex in general we have to start from a reasonable initial estimate if we expect a standard nonlinear optimization algorithm to converge to the global optimum. Fortunately, SLAM is an incremental process, and in every step we only add a small number of new variables for which we have locally very accurate measurements available. However, it is always possible to find a scenario in which this assumption will not hold. If the sensors are very noisy, for example, we might get stuck in a local optimum. While I have not encountered such data in my work, there might be

a solution based on work by Olson *et al.* [113], which represents a batch SLAM algorithm that uses a local parametrization to avoid global minima.

Another assumption I make is that application of the variable ordering heuristic COLAMD results in a sparse factor matrix. As discussed in Section 3.6, for some special cases of SLAM applications no variable order exists that produces a sparse factor matrix, as the original information matrix already contains dense blocks. But beyond these special cases it is not clear if there are SLAM settings for which a sparse factor matrix exists, but for which COLAMD produces a suboptimal ordering that leads to a dense factor matrix.

It is important to point out that I do not have to make assumptions about the linearity of the measurement functions in order to guarantee consistency of the solution. Because iSAM does not marginalize out any variables, it can and does relinearize measurement equations based on improved variable estimates. iSAM is equivalent to full nonlinear optimization, even though the iterations are spread over multiple steps. Therefore, iSAM correctly deals with nonlinear measurement functions, that is it converges to the correct solution even for nonlinear measurements under the assumption of a locally convex error function discussed above.

Do we have to assume the availability of range measurements, or can iSAM deal with bearing-only measurements? iSAM implements sparse bundle adjustment, a technique commonly used in the structure-from-motion community to perform 3D reconstructions from images only, including determining the camera poses and optionally their calibration. However, not having range available raises the issue of how to initialize a new landmark, as a single measurement is not sufficient to provide any depth information. Before initializing a new landmark, one can wait until at least two measurements with a sufficient baseline become available, providing a reasonable initialization for the landmark. A better approach is to reflect the lack of range information in the parametrization of the landmarks. One example of such work is the so-called measurement subspace by Folkesson *et al.* [48].

6.3 Contributions

The major contribution of this work is the *exact incremental solution* to the SLAM problem in real-time based on a direct equation solver. Direct equation solvers are preferable to other solutions because of their quadratic convergence rate. Convergence to the correct solution of course assumes that a good initial estimate is available, but that can be expected for incremental SLAM applications. While many have discarded direct equation solvers as too expensive for real-time SLAM applications, my work clearly shows that direct equation solvers provide a very competitive solution when one understands how to apply them correctly in the SLAM context. The key to efficiency is to exploit the sparsity of the problem, and to ensure that the resulting factor matrix remains sparse during the estimation process by ordering the variables in a suitable way. The combination of factorized smoothing information matrix with variable ordering and incremental factorization updates using Givens rotations makes my real-time direct solver based solution to SLAM possible, and I have shown that it compares favorably with other methods, especially also methods based on iterative equation solvers.

A second major contribution of this work is to provide *efficient access to marginal covariances*. It would seem that the EKF is suitable for this task as it directly operates on the covariance matrix. However, for nonlinear SLAM problems, the EKF provides inconsistent results and is therefore not a useful approach to obtaining estimation uncertainties. Also, by explicitly representing the dense covariance matrix the EKF severely restricts the size of problems it can handle. Using the information form instead allows for efficient as well as exact solutions to SLAM. However, typical exact solutions based on the information form are based on iterative equation solvers which do not allow access to the covariances without simply inverting the complete information matrix, an operation that is infeasible for any non-trivial application. That is where my approach shows its advantages: By using the factorized information matrix, I exploit the sparsity of the information matrix, while at the same time allowing efficient access to any parts of its inverse, the covariance matrix. Calculating all entries of this dense covariance matrix would again render the approach useless, because their number scales with the square of the number of variables. Instead I

have shown how to selectively calculate entries of interest without having to calculate the complete dense matrix.

Introducing methods from the sparse linear algebra literature into the field of SLAM can be seen as another contribution. While matrix factorizations have been used before in other contexts, updating of matrix factorizations for SLAM estimation has not been presented before. In particular the combination of matrix factorization, variable reordering and incremental factorization updates for efficiency is novel and opens new possibilities that start to be exploited in other robotics applications, such as [119].

I present different *state of the art data association techniques* and show how they work in connection with iSAM. In particular, I examine which components of the covariance matrix are required in order to perform these various methods. Most interesting is the joint compatibility branch and bound (JCBB) algorithm by Neira and Tardos [105] that has proven very successful in SLAM applications. It is successful because it takes into account the coupling between multiple measurements, making it very robust against wrong matches. Wrong matches in SLAM often lead to failures that an estimation algorithm cannot recover from, therefore being able to use JCBB is a major advantage of iSAM over many other SLAM algorithms.

I approach the problem of *which measurements are useful* in the estimation process by applying an information theoretic approach pioneered by Davison [22]. Not all measurements provide useful information, and simply adding all potentially available measurements only complicates the estimation problem without necessarily improving the quality of the result. I found that the same parts of the covariance matrix are required as for JCBB, which makes those methods complimentary in SLAM applications. I present results on how omitting measurements affects the computational complexity.

I finally show that *iSAM successfully works in general SLAM settings* including visual SLAM. I show experiments for settings including estimation problems such as landmark based and pose-only, different environments such as indoor and outdoor, as well as different sensor modalities, such as laser range scanners and cameras. In particular, visual SLAM

in unstructured outdoor environments provides a very challenging test for any SLAM algorithm, because range information is very uncertain and data association very ambiguous. Previous algorithms have either been restricted to a very small number of landmarks, or if they tracked many landmarks they were not able to close loops. iSAM combines advantages from both methods, even though there are still many improvements possible towards a truly robust visual SLAM system.

6.4 Sensor Calibration

While sensor calibration can be included in the estimation process, it is generally easier to separately calibrate before the actual application. Sensor calibration is performed by adding a small number of variables into the estimation process. These might be as simple as an offset of a laser range finder on the vehicle, or full 6 DOF sensor pose including rotations, and can also extend to intrinsic camera parameters such as focal lengths and principal point. Typically a prior is used as the approximate parameters are known, for example by measuring the sensor's location on the vehicle, or from the data sheet of a camera. A prior is sometimes also needed for weakly constraint parameters such as the principal point of a camera.

iSAM directly supports sensor calibration, but the estimation problem naturally becomes more expensive to solve. The additional measurement equations necessary for calibration can be added like any other measurement. However, the same calibration variables are used continuously, creating relatively direct constraints between all pose variables of the estimation problem. This leads to increased fill-in, and therefore makes the estimation process more expensive. However, the smoothing information matrix still remains sparse except for one dense block column that will also be reflected in the R factor.

Typically a good calibration is obtained from a short sequence, assuming a relatively general motion of the vehicle. Therefore, when done incrementally, calibration would be stopped after a certain time, and normal estimation as presented in this work continues. This, however, is equivalent to first estimating the calibration, and then actually running the application.

6.5 Future Work

At which time should the batch reordering be performed? My current solution is to reorder after a constant number of steps, where that constant might be adjusted depending on the nature of the specific SLAM problem. Instead, one can imagine to monitor the fill-in that arises from incrementally adding new measurements, and to automatically determine the best time for a batch reordering step. There will need to be a balance between the cost of reordering and the increased cost of updates based on growing fill-in.

Can one modify the variable ordering so that adding new measurements becomes cheaper? My current batch variable reordering based on the COLAMD heuristic targets the variable ordering that produces the lowest fill-in for the current matrix. However, it does not take into account that more measurements and new variables will be added at the end of the current trajectory. In particular, COLAMD might move the previous pose to the left side of the matrix, which will produce a dense column when the next pose is added because the odometry measurement has to connect both poses. It might be possible to slightly modify the COLAMD ordering without incurring much additional fill-in, but making incremental updates much cheaper as a consequence. Such an ordering is used by Frese [54] in a tree-based data structure, however, the overall algorithm performs approximations to reach that goal.

If we extend this thought it leads to the question if an *incremental variable ordering* exists that produces low fill-in. I currently add new variables at the end of the R factor, but I could also insert them anywhere in the middle of the matrix. However, it is not clear if a SLAM specific incremental ordering can be found, because this might require knowledge about when and where loops will be closed in the future. An alternative is to modify the variable ordering incrementally by changing the position of individual variables. Removing variables to insert them somewhere else requires QR-downdating [60] of the factor matrix in addition to updating, which I have not explored yet.

One could also perform *incremental relinearization*, which in combination with incremental variable ordering would lead to an algorithm that is completely free of batch steps.

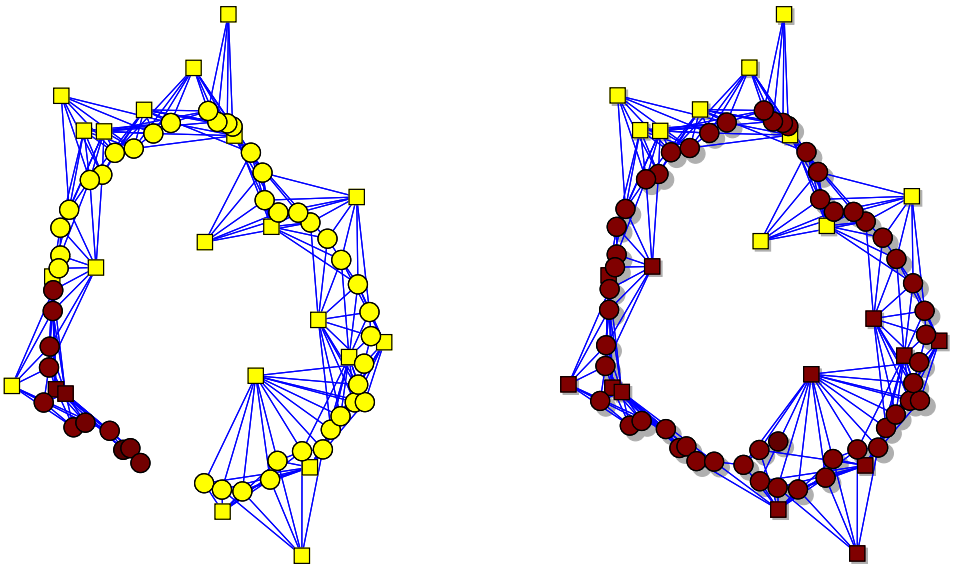


Figure 49: The Wildfire algorithm [118] only updates nodes which significantly change, starting from the most recently added node. In normal operation (left) only the most recently observed nodes are updated. But during loop closings (right) most of the variables require an update. Updated variables are shown in brown (dark), the remaining ones in yellow (light).

Relinearization is typically only required for a small number of variables, because measurements are local and only affect a small number of variables directly, with their effects rapidly declining while propagating through the constraint graph. An exception is a situation such as a loop closing event, which can affect many variables at once. But in most cases it is sufficient to perform selective relinearization only over variables whose estimate has changed by more than some threshold. While I have not implemented this yet, the solution is to remove first the affected measurement rows from the current factor by QR-downdating, and then adding the relinearized measurement rows by QR-updating as described earlier.

We can also *reduce the cost for back-substitution* by only calculating entries that are actually needed. A constant time approximation can be obtained in most cases, as only the most recent variables change significantly enough to warrant recalculation. Back-substitution can for example be stopped once the change in the variable estimate drops below a threshold, which is the matrix equivalent to the wild-fire algorithm in Loopy SAM [118], see Figure 49. It is expected that this constant time approximation would be sufficient for most cases, and the full calculation would only become necessary when large loops are closed.

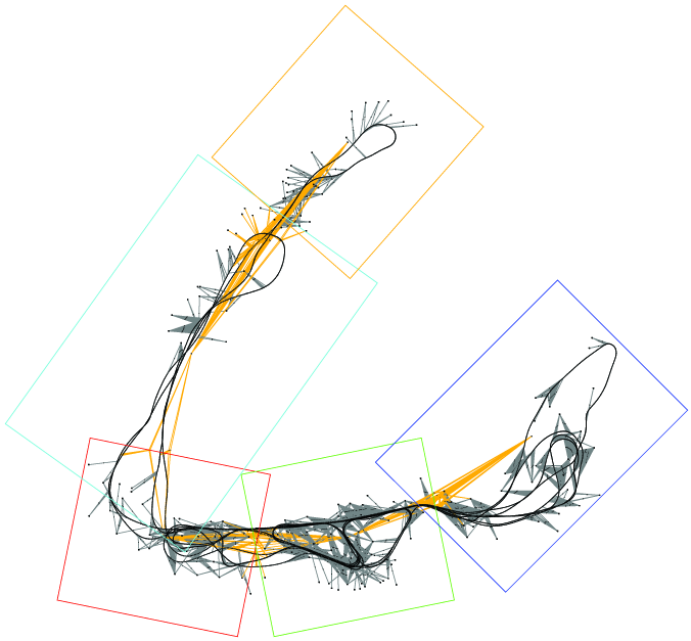


Figure 50: Out-of-core submap-based variant of SAM applied to the Victoria Park dataset (from [107]).

How can we *deal with the ever increasing number of variables* due to the inclusion of the complete robot trajectory in the smoothing formulation? I have already addressed how to reduce the number of measurements to simplify the estimation problem, by analyzing the information provided by specific measurements. For the number of poses the problem is somewhat different as the robot trajectory is represented by a chain. One potential solution is to combine multiple poses, similar to the star nodes in Graphical SLAM [46]. A further idea to deal with increasing complexity over time is to split the problem into multiple subproblems. A common approach is to use submaps, however, these approaches typically make approximations that we would like to avoid. An exact batch solution to submapping in connection with SAM has been presented in [107], with an example shown in Figure 50.

When a *robot operates continuously* in the same environment, a different solution needs to be found to bound the growth of the estimation problem over time. One way is to stop mapping once a certain quality of the state estimate is reached, and then continue with robot localization only based on the estimated map. The decision of when to stop might be based on the uncertainty remaining in the system, or on a measure of how much

information new measurements would add to the estimate. Following this thought further leads to a system that only adds new robot poses when a measurement is encountered that warrants inclusion into the map because it has not been seen before or because it contradicts information already contained in the map.

A significant improvement in the implementation of iSAM is expected from using *better data structures* targeted towards large-scale problems. In particular, the sparse matrix implementation requires careful consideration of which operations need to be supported efficiently. An equivalent formulation to iSAM could be based on graphs instead of sparse matrices. Given enough memory, it is expected that iSAM could deal with two to three orders of magnitude larger data-sets than presented here. At some point however, memory (and the batch steps I currently perform) will become the limiting factor. One possibility then is to perform out-of-core computations similar to the batch SAM algorithm presented in [107].

In addition to adding measurements at any time, iSAM should also be able to remove them as needed. Again, downdating is necessary to remove measurements, and potential applications include reversing data association decisions when they are later found to be incorrect, and removing earlier parts of the trajectory for example to keep the size of the estimation problem constant. I also have not addressed adding measurements to earlier parts of the system, which is for example necessary for delayed data association techniques that allow more robust decisions about when a loop closing occurs.

Extending iSAM to a *distributed SLAM* algorithm would allow multi-robot mapping as well as exploiting multi-core processors. A good starting point might be the batch solution of multi-frontal QR factorization for SAM [30], with a simple example shown in Figure 51. Especially the recovery of marginal covariances and data association would need to be addressed.

My work currently does not deal with *dynamic scenes*. Moving objects need to be identified as they will otherwise lead to inconsistencies in the SLAM estimate by introducing constraints that conflict with the static components of the scene. Dealing with moving objects is currently an active area of research, and existing and future methods might be

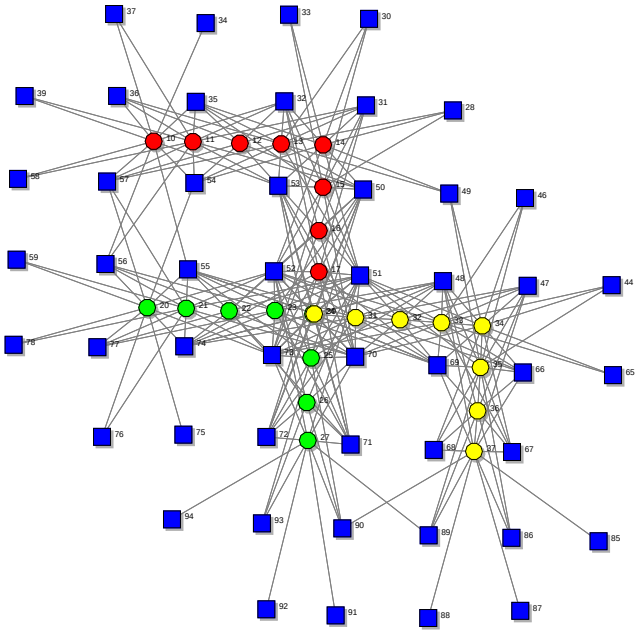


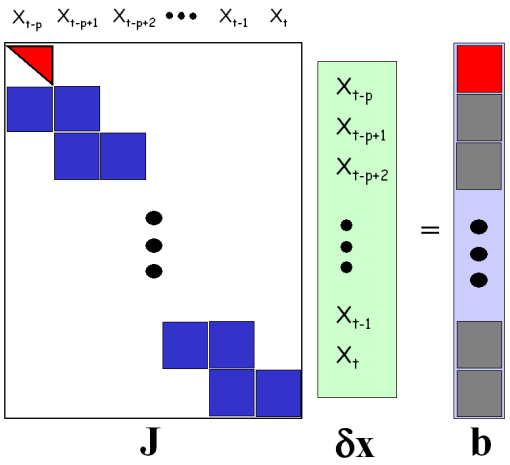
Figure 51: Multi-robot mapping using multi-frontal QR factorization for SAM (from [30]). Landmarks are shown as squares, robot poses as circles, and the three different robots are identified by different colors.

applicable to iSAM. JCBB data association helps to some degree in avoiding negative impact of moving objects as long as most of the scene is static.

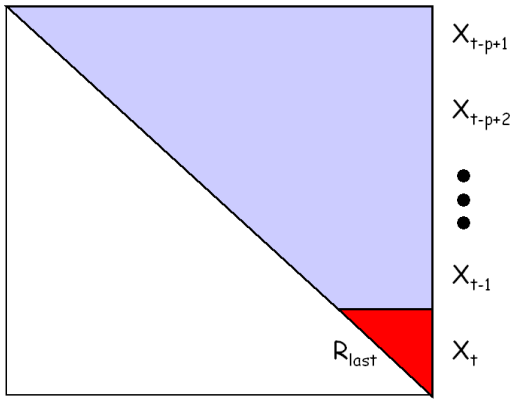
The real-time properties of iSAM provide data necessary for autonomous operation of mobile platforms. I imagine commercial applications of my work on consumer mobile robot products, but it would also be useful when using robots for mapping buildings or entire cities for virtual reality applications.

Significant portions of this work arose from looking at both the graphical model underlying the SLAM problem, as well as the sparse linear algebra formulation. The combination of different fields is typically a very fruitful approach, and including other fields might provide novel and better solutions. But there is still also potential in formulating iSAM completely as a problem on graphs rather than matrices.

There are applications of this work beyond robotic SLAM. One interesting application is tracking a sensor in unknown settings for augmented reality, as a cheap alternative to instrumenting the environment. My visual SLAM applications of iSAM benefits from scalable and exact solutions, especially for unstructured outdoor environments.



(a) Linearized fixed-lag smoothing system.



(b) The R factor obtained when solving the system on the left.

Figure 52: Our fixed-lag smoothing algorithm [117] based on square root information smoothing. The prior distribution on $\mathbf{x}_{t-p}$ in (a) is obtained by remembering R_{last} from (b) from solving the system p steps earlier.

Incrementally adding new measurements to a factorized representation of a Gaussian distribution is generally of interest in a number of areas. Ranganathan and Yang for example applied our method to online Gaussian process estimation [119] and used the technique for visual tracking of objects. And Folkesson [49] shows in concurrent work how to track features in underwater applications using incremental QR updates.

Another application of incremental updates is fixed-lag smoothing. We have presented a fixed-lag smoother for pose estimation with out-of-sequence measurements [117], see Figure 52. While the problem was handled in a different way, for more expensive estimation problems the incremental solution provided by iSAM can become necessary.

Instead of fixed-lag smoothing, as commonly used to improve the quality of visual odometry, one can also imagine to use full incremental smoothing. In fact, the solution will be constant time because of the absence of loops, as long as only a constant number of most recent variables are recovered. However, when needed, the full optimization problem is available at the cost of storing and maintaining the necessary data structures.

My efficient marginal covariance recovery algorithm is more generally applicable than just in the context of iSAM. Whenever a square root information matrix is available or can be calculated efficiently, such as in the original batch SAM work, my algorithm will allow

efficient access to components of the covariance matrix without calculating the complete matrix.

6.6 Final Thoughts

iSAM improves upon the state of the art by providing a superior approach to SLAM that is exact, is efficient, supports data association, and is generally applicable to a wide range of SLAM problems. I hope it will prove useful as a basis for future research and that it will eventually find its way into commercial applications. It could well be a step on our way to a future in which robots are omnipresent and help us in everyday life.

REFERENCES

[1] Arya, S., Mount, D., Netanyahu, N., Silverman, R., and Wu, A., "An optimal algorithm for approximate nearest neighbor searching," *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

[2] Bailey, T. and Durrant-Whyte, H., "Simultaneous localisation and mapping (SLAM): Part II state of the art," *Robotics & Automation Magazine*, Sep 2006.

[3] Baltzakis, H. and Trahanias, P., "An iterative approach for building feature maps in cyclic environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 576–581, 2002.

[4] Baltzakis, H. and Trahanias, P., "Closing multiple loops while mapping features in cyclic environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 717–722, 2003.

[5] Bar-Shalom, Y. and Li, X., *Estimation and Tracking: principles, techniques and software*. Boston, London: Artech House, 1993.

[6] Beardsley, P., Torr, P., and Zisserman, A., "3D model acquisition from extended image sequences," in *Eur. Conf. on Computer Vision (ECCV)*, pp. II:683–695, 1996.

[7] Besl, P. and McKay, N., "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, 1992.

[8] Bierman, G., *Factorization methods for discrete sequential estimation*, vol. 128 of *Mathematics in Science and Engineering*. New York: Academic Press, 1977.

[9] Bolles, R. and Fischler, M., "A RANSAC-based approach to model fitting and its application to finding cylinders in range data," in *Intl. Joint Conf. on AI (IJCAI)*, (Vancouver, BC, Canada), pp. 637–643, 1981.

[10] Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., and Teller, S., "An Atlas framework for scalable mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1899–1906, Sep 2003.

[11] Bosse, M., Newman, P., Leonard, J., and Teller, S., "Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework," *Intl. J. of Robotics Research*, vol. 23, pp. 1113–1139, Dec 2004.

[12] Bunschoten, R. and Kröse, B., "Visual odometry from an omnidirectional vision system," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, pp. 577–583, 2003.

[13] Chum, O., Werner, T., and Matas, J., "Two-view geometry estimation unaffected by a dominant plane," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[14] Clemente, L., Davison, A., Reid, I., Neira, J., and Tardós, J., "Mapping large loops with a single hand-held camera," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[15] Corke, P., Strelow, D., and Singh, S., "Omnidirectional visual odometry for a planetary rover," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2004.

[16] Cover, T. and Thomas, J., *Elements of Information Theory*. John Wiley & Sons, New York, NY, 1991.

[17] Davis, T. and Hager, W., "Modifying a sparse Cholesky factorization," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 3, pp. 606–627, 1996.

[18] Davis, T., *Direct Methods for Sparse Linear Systems*. Fundamentals of Algorithms, Society for Industrial and Applied Mathematics, 2006.

[19] Davis, T., Gilbert, J., Larimore, S., and Ng, E., "A column approximate minimum degree ordering algorithm," *ACM Trans. Math. Softw.*, vol. 30, no. 3, pp. 353–376, 2004.

[20] Davison, A., "SLAM with a single camera," in *SLAM/CML Workshop at ICRA 2002*, May 2002.

[21] Davison, A., "Real-time simultaneous localisation and mapping with a single camera," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 1403–1410, Oct 2003.

[22] Davison, A., "Active search for real-time vision," in *Intl. Conf. on Computer Vision (ICCV)*, Oct 2005.

[23] Davison, A., Cid, Y., and Kita, N., "Real-time 3D SLAM with wide-angle vision," in *5th IFAC/EURON Symp. on Intelligent Autonomous Vehicles, IAV'04*, Jul 2004.

[24] Davison, A. and Kita, N., "3D simulataneous localisation and map-building using active vision for a robot moving on undulating terrain," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Dec 2001.

[25] Davison, A., Mayol, W., and Murray, D., "Real-time localisation and mapping with wearable active vision," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, Oct 2003.

[26] Davison, A. and Murray, D., "Simulataneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 865–880, Jul 2002.

[27] Davison, A., Reid, I., Molton, N., and Stasse, O., "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, pp. 1052–1067, Jun 2007.

[28] Dellaert, F., *Monte Carlo EM for Data Association and its Applications in Computer Vision*. PhD thesis, School of Computer Science, Carnegie Mellon, September 2001. Also available as Technical Report CMU-CS-01-153.

[29] Dellaert, F., "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, 2005.

[30] Dellaert, F., Kipp, A., and Krauthausen, P., "A multifrontal QR factorization approach to distributed inference applied to multi-robot localization and mapping," in *Proc. 22^{nd} AAAI National Conference on AI*, (Pittsburgh, PA), 2005.

[31] Dellaert, F., Seitz, S., Thorpe, C., and Thrun, S., "EM, MCMC, and chain flipping for structure from motion with unknown correspondence," *Machine learning*, vol. 50, pp. 45–71, January - February 2003. Special issue on Markov chain Monte Carlo methods.

[32] Dellaert, F. and Kaess, M., "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.

[33] Dennis, J. and Schnabel, R., *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, 1983.

[34] Duckett, T., Marsland, S., and Shapiro, J., "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002.

[35] Durrant-Whyte, H., "Uncertain geometry in robotics," *IEEE Trans. Robot. Automat.*, vol. 4, no. 1, pp. 23–31, 1988.

[36] Durrant-Whyte, H. and Bailey, T., "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics & Automation Magazine*, Jun 2006.

[37] Eade, E. and Drummond, T., "Scalable monocular SLAM," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Jun 2006.

[38] Eliazar, A. and Parr, R., "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[39] Elinas, P., Sim, R., and Little, J., "σSLAM: Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006.

[40] Engels, C., Stewénius, H., and Nistér, D., "Bundle adjustment rules," in *Symposium on Photogrammetric Computer Vision*, Sep 2006.

[41] Eustice, R., Singh, H., and Leonard, J., "Exactly sparse delayed-state filters," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2417–2424, April 2005.

[42] Eustice, R., Singh, H., Leonard, J., Walter, M., and Ballard, R., "Visually navigating the RMS titanic with SLAM information filters," in *Robotics: Science and Systems (RSS)*, Jun 2005.

[43] Eustice, R., Singh, H., and Leonard, J., "Exactly sparse delayed-state filters for view-based slam," *IEEE Trans. Robotics*, vol. 22, pp. 1100–1114, Dec 2006.

[44] Eustice, R., Singh, H., Leonard, J., and Walter, M., "Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters," *Intl. J. of Robotics Research*, vol. 25, pp. 1223–1242, Dec 2006.

[45] Fischler, M. and Bolles, R., "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography," *Commun. Assoc. Comp. Mach.*, vol. 24, pp. 381–395, 1981.

[46] Folkesson, J. and Christensen, H., "Graphical SLAM - a self-correcting map," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, pp. 383–390, 2004.

[47] Folkesson, J. and Christensen, H., "Closing the loop with graphical SLAM," *IEEE Trans. Robotics*, vol. 23, pp. 731–741, Aug 2007.

[48] Folkesson, J., Jensfelt, P., and Christensen, H., "Vision SLAM in the measurement subspace," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Apr 2005.

[49] Folkesson, J., Leonard, J., Leederkerken, J., and Williams, R., "Feature tracking for underwater navigation using sonar," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3678–3684, Oct 2007.

[50] Frahm, J. and Pollefeys, M., "RANSAC for (quasi-)degenerate data (QDEGSAC)," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[51] Frese, U., *An $O(\log n)$ Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments*. PhD thesis, University of Erlangen-Nürnberg, 2004.

[52] Frese, U., "Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[53] Frese, U., Larsson, P., and Duckett, T., "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. Robotics*, vol. 21, pp. 196–207, April 2005.

[54] Frese, U. and Schröder, L., "Closing a million-landmarks loop," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 5032–5039, Oct 2006.

[55] Gaspar, J., Winters, N., and Santos-Victor, J., "Vision-based navigation and environmental representations with an omnidirectional camera," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 890–898, Dec 2000.

[56] Gauss, C., *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Mabientium [Theory of the Motion of the heavenly Bodies Moving about the Sun in Conic Sections]*. Hamburg, Germany: Perthes and Besser, 1809. English translation available at http://name.umdl.umich.edu/AGG8895.0001.001.

[57] Gauss, C., "Disquisitio de elementis ellipticis Palladis [Disquisition on the elliptical elements of Pallas]," *Göttingische gelehrte Anzeigen*, December 1810.

[58] Gentleman, W., "Least squares computations by Givens transformations without square roots," *IMA J. of Appl. Math.*, vol. 12, pp. 329–336, 1973.

[59] Gill, P., Golub, G., Murray, W., and Saunders, M., "Methods for modifying matrix factorizations," *Mathematics and Computation*, vol. 28, no. 126, pp. 505–535, 1974.

[60] Golub, G. and Loan, C. V., *Matrix Computations*. Baltimore: Johns Hopkins University Press, third ed., 1996.

[61] Golub, G. and Plemmons, R., "Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition," *Linear Algebra and Its Applications*, vol. 34, pp. 3–28, Dec 1980.

[62] Griewank, A., "On Automatic Differentiation," in *Mathematical Programming: Recent Developments and Applications* (Iri, M. and Tanabe, K., eds.), pp. 83–108, Kluwer Academic Publishers, 1989.

[63] Grisetti, G., Stachniss, C., and Burgard, W., "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 667–672, 2005.

[64] Grisetti, G., Stachniss, C., and Burgard, W., "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[65] Grisetti, G., Stachniss, C., Grzonka, S., and Burgard, W., "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[66] Guivant, J. and Nebot, E., "Optimization of the simultaneous localization and map building algorithm for real time implementation," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 242–257, June 2001.

[67] Gutmann, J.-S. and Konolige, K., "Incremental mapping of large cyclic environments," in *IEEE Intl. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 318–325, 1999.

[68] Gutmann, J.-S. and Nebel, B., "Navigation mobiler roboter mit laserscans," in *Autonome Mobile Systeme*, (Berlin), Springer Verlag, 1997.

[69] Hähnel, D., Burgard, W., Fox, D., and Thrun, S., "A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 206–211, 2003.

[70] Hähnel, D., Burgard, W., Wegbreit, B., and Thrun, S., "Towards lazy data association in SLAM," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, (Sienna, Italy), Springer, 2003.

[71] Harris, C. and Stephens, M., "A combined corner and edge detector," *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, August 1988.

[72] Harris, C. and Pike, J., "3D positional integration from image sequences," *Image and Vision Computing*, vol. 6, pp. 87–90, May 1988.

[73] Hartley, R. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[74] Jonker, R. and Volgenant, A., "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[75] Julier, S. and Uhlmann, J., "A counter example to the theory of simultaneous localization and map building," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 4, pp. 4238–4243, 2001.

[76] Kaess, M., Arkin, R., and Rossignac, J., "Compact encoding of robot-generated 3D maps for efficient wireless transmission," in *IEEE Intl. Conf. on Advanced Robotics (ICAR)*, (Coimbra, Portugal), pp. 324–331, 2003.

[77] Kaess, M. and Dellaert, F., "A Markov chain Monte Carlo approach to closing the loop in SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Barcelona, Spain), pp. 645–650, April 2005.

[78] Kaess, M. and Dellaert, F., "Visual SLAM with a multi-camera rig," Tech. Rep. GIT-GVU-06-06, Georgia Institute of Technology, Feb 2006.

[79] Kaess, M., Ranganathan, A., and Dellaert, F., "Fast incremental square root information smoothing," in *Intl. Joint Conf. on AI (IJCAI)*, (Hyderabad, India), pp. 2129–2134, 2007.

[80] Kaess, M., Ranganathan, A., and Dellaert, F., "iSAM: Fast incremental smoothing and mapping with efficient data association," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Rome, Italy), pp. 1670–1677, April 2007.

[81] Kaess, M., Ranganathan, A., and Dellaert, F., "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, Dec 2008.

[82] Karlsson, N., Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., and Munich, M., "The vSLAM algorithm for robust localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 24–29, 2005.

[83] Karlsson, N., Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., and Munich, M., "The vSLAM algorithm for robust localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 24–29, Apr 2005.

[84] Khan, Z., Balch, T., and Dellaert, F., "MCMC data association and sparse factorization updating for real time multitarget tracking with merged and multiple measurements," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 28, pp. 1960–1972, December 2006.

[85] Knight, J., Davison, A., and Reid, I., "Towards constant time SLAM using postponement," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 405–413, 2001.

[86] Konolige, K., "Large-scale map-making," in *Proc. 21th AAAI National Conference on AI*, (San Jose, CA), 2004.

[87] Konolige, K. and Agrawal, M., "Frame-frame matching for realtime consistent visual mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 2803–2810, Apr 2007.

[88] Konolige, K., Agrawal, M., Bolles, R., Cowan, C., Fischler, M., and Gerkey, B., "Outdoor mapping and navigation using stereo vision," in *Intl. Sym. on Experimental Robotics (ISER)*, Jul 2006.

[89] Konolige, K., Agrawal, M., and Sola, J., "Large scale visual odometry for rough terrain," in *IROS visual SLAM workshop*, Oct 2007.

[90] Krauthausen, P., Dellaert, F., and Kipp, A., "Exploiting locality by nested dissection for square root smoothing and mapping," in *Robotics: Science and Systems (RSS)*, 2006.

[91] Leonard, J., Durrant-Whyte, H., and Cox, I., "Dynamic map building for an autonomous mobile robot," *Intl. J. of Robotics Research*, vol. 11, no. 4, pp. 286–289, 1992.

[92] Levin, A. and Szeliski, R., "Visual odometry and map correlation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[93] Ling, F., "Givens rotation based least squares lattice related algorithms," *IEEE Trans. Signal Processing*, vol. 39, pp. 1541–1551, Jul 1991.

[94] Lu, F. and Milios, E., "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, pp. 333–349, Apr 1997.

[95] Marks, T., Howard, A., Bajracharya, M., Cottrell, G., and Matthies, L., "Gamma-SLAM: Using stereo vision and variance grid maps for SLAM in unstructured envrionments," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2008.

[96] Maybeck, P., *Stochastic Models, Estimation and Control*, vol. 1. New York: Academic Press, 1979.

[97] Merwe, R. and Wan, E., "The square-root unscented Kalman filter for state and parameter estimation," in *Intl. Conf. Acoust., Speech, and Signal Proc. (ICASSP)*, pp. 3461–3464, 2001.

[98] Montemerlo, M. and Thrun, S., "Simultaneous localization and mapping with unknown data association using FastSLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[99] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. 19th AAAI National Conference on AI*, (Edmonton, Alberta, Canada), 2002.

[100] Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B., "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[101] Montiel, J., Civera, J., and Davison, A., "Unified inverse depth parametrization for monocular SLAM," in *Robotics: Science and Systems (RSS)*, Aug 2006.

[102] Mottaghi, R., Kaess, M., Ranganathan, A., Roberts, R., and Dellaert, F., "Place recognition-based fixed-lag smoothing for environments with unreliable GPS," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Pasadena, CA), May 2008.

[103] Murray, D. and Jennings, C., "Stereo vision based mapping and navigation for mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1694–1699, Apr 1997.

[104] Neira, J., Rbeiro, M., and Tardós, J., "Mobile robot localisation and map building using monocular vision," in *Intl. Symp. on Intelligent Robotics Systems*, Jul 1997.

[105] Neira, J. and Tardos, J., "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 890–897, December 2001.

[106] Ni, K. and Dellaert, F., "Stereo tracking and three-point/one-point algorithms - a robust approach in visual odometry," in *Intl. Conf. on Image Processing (ICIP)*, 2006.

[107] Ni, K., Steedly, D., and Dellaert, F., "Tectonic SAM: Exact; out-of-core; submap-based SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Rome; Italy), April 2007.

[108] Nieto, J., Guivant, H., Nebot, E., and Thrun, S., "Real time data association for FastSLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2003.

[109] Nistér, D., "Preemptive RANSAC for live structure and motion estimation," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 199–206, Oct 2003.

[110] Nistér, D., Naroditsky, O., and Bergen, J., "Visual odometry," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 652–659, Jun 2004.

[111] Nistér, D., Naroditsky, O., and Bergen, J., "Visual odometry for ground vehicle applications," *J. of Field Robotics*, vol. 23, Jan 2006.

[112] Olson, C., "Stereo ego-motion improvements for robust rover navigation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1099–1104, 2001.

[113] Olson, E., Leonard, J., and Teller, S., "Fast iterative alignment of pose graphs with poor initial estimates," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006.

[114] Olson, E., Leonard, J., and Teller, S., "Spatially-adaptive learning rates for online incremental SLAM," in *Robotics: Science and Systems (RSS)*, Jun 2007.

[115] Paskin, M., "Thin junction tree filters for simultaneous localization and mapping," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[116] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes: The Art of Scientific Computing*. Cambridge (UK) and New York: Cambridge University Press, 2nd ed., 1992.

[117] Ranganathan, A., Kaess, M., and Dellaert, F., "Fast 3D pose estimation with out-of-sequence measurements," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (San Diego, CA), Oct 2007. To appear.

[118] Ranganathan, A., Kaess, M., and Dellaert, F., "Loopy SAM," in *Intl. Joint Conf. on AI (IJCAI)*, (Hyderabad, India), pp. 2191–2196, 2007.

[119] Ranganathan, A. and Yang, M.-H., "Online sparse matrix Gaussian process regression and vision applications," in *Eur. Conf. on Computer Vision (ECCV)*, pp. 468–482, 2008.

[120] Se, S., Lowe, D., and Little, J., "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *Intl. J. of Robotics Research*, vol. 21, pp. 735–758, Aug 2002.

[121] Se, S., Lowe, D., and Little, J., "Vision-based global localization and mapping for mobile robots," *IEEE Trans. Robotics*, vol. 21, pp. 364–375, Jun 2005.

[122] Sim, R., Elinas, P., Griffin, M., and Little, J., "Vision-based SLAM using the Rao-Blackwellised particle filter," in *Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, Jul 2005.

[123] Sim, R., Elinas, P., Griffin, M., Shyr, A., and Little, J., "Design and analysis of a framework for real-time vision-based SLAM using Rao-Blackwellised particle filters," in *Proc. of the 3rd Canadian Conf. on Computer and Robotic Vision (CRV)*, Jun 2006.

[124] Sim, R. and Little, J., "Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2006.

[125] Smith, P., Reid, I., and Davison, A., "Real-time monocular SLAM with straight lines," in *British Machine Vision Conf. (BMVC)*, Sep 2006.

[126] Smith, R. and Cheeseman, P., "On the representation and estimation of spatial uncertainty," *Intl. J. of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987.

[127] Smith, R., Self, M., and Cheeseman, P., "A stochastic map for uncertain spatial relationships," in *Int. Symp on Robotics Research*, 1987.

[128] Smith, R., Self, M., and Cheeseman, P., "Estimating uncertain spatial relationships in Robotics," in *Autonomous Robot Vehicles* (Cox, I. and Wilfong, G., eds.), pp. 167–193, Springer-Verlag, 1990.

[129] Stasse, O., Davison, A., Sellaouti, R., and Yokoi, K., "Real-time 3D SLAM for humanoid robot considering pattern generator information," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct 2006.

[130] Thrun, S., "Robotic mapping: a survey," in *Exploring artificial intelligence in the new millennium*, pp. 1–35, Morgan Kaufmann, Inc., 2003.

[131] Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[132] Thrun, S., Fox, D., and Burgard, W., "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine learning*, vol. 31, pp. 29–53, 1998.

[133] Thrun, S. and Liu, Y., "Multi-robot SLAM with sparse extended information filters," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, (Sienna, Italy), Springer, 2003.

[134] Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A., "Bundle adjustment – a modern synthesis," in *Vision Algorithms: Theory and Practice* (Triggs, W., Zisserman, A., and Szeliski, R., eds.), LNCS, pp. 298–375, Springer Verlag, Sep 1999.

[135] Tucakov, V., Sahota, M., Murray, D., Mackworth, A., Little, J., Kingdon, S., C.Jennings, and Barman, R., "Spinoza: A stereoscopic visually guided mobile robot," in *Proc. of Hawaii Intl. Conf. on Systems Sciences*, vol. 5, pp. 188–197, Jan 1997.

[136] Wang, Z., *Exactly Sparse Information Filters for Simultaneous Localization and Mapping*. PhD thesis, The University of Technology, Sydney, 2007.

[137] Wang, Z., Huang, S., and Dissanayake, G., "DSLAM: Decoupled localization and mapping for autonomous robots," in *International Symposium of Robotics Research (ISRR 05)*, Oct 2005.

[138] Zhu, Z., Oskiper, T., Samarasekera, S., Kumar, R., and Sawhney, H., "Ten-fold improvement in visual odometry using landmark matching," in *Intl. Conf. on Computer Vision (ICCV)*, Oct 2007.

INDEX